



SRC-PE Software

Network Guide: SAE, Juniper Networks Routers, and NIC

Release 1.0.x

Juniper Networks, Inc.

1194 North Mathilda Avenue
Sunnyvale, CA 94089

USA

408-745-2000

www.juniper.net

Juniper Networks, the Juniper Networks logo, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. JUNOS and JUNOSe are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

This product includes the following software: Fontconfig, X FreeType library, X Render extension headers, and X Render extension library, copyright © 2001, 2003 Keith Packard.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Keith Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Keith Packard makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

KEITH PACKARD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL KEITH PACKARD BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Products made or sold by Juniper Networks (including the ERX-310, ERX-705, ERX-710, ERX-1410, ERX-1440, M5, M7i, M10, M10i, M20, M40, M40e, M160, M320, and T320 routers, T640 routing node, and the JUNOS, JUNOSe, and SDX-300 software) or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Copyright © 2007, Juniper Networks, Inc.
All rights reserved. Printed in USA.

SRC-PE Software Network Guide: SAE, Juniper Networks Routers, and NIC, Release 1.0.x
Writing: Linda Creed, Justine Kangas, Betty Lew, Helen Shaw
Editing: Fran Mues
Illustration: Nathaniel Woodward
Cover Design: Edmonds Design

Revision History
6 April 2007—Revision 1

The information in this document is current as of the date listed in the revision history.

Software License

The terms and conditions for using this software are described in the software license contained in the acknowledgment to your purchase order or, to the extent applicable, to any reseller agreement or end-user purchase agreement executed between you and Juniper Networks. By using this software, you indicate that you understand and agree to be bound by those terms and conditions.

Generally speaking, the software license restricts the manner in which you are permitted to use the software and may contain prohibitions against certain uses. The software license may state conditions under which the license is automatically terminated. You should consult the license for further details.

For complete product documentation, please see the Juniper Networks Web site at www.juniper.net/techpubs.

END USER LICENSE AGREEMENT

READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. The Parties. The parties to this Agreement are Juniper Networks, Inc. and its subsidiaries (collectively "Juniper"), and the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software ("Customer") (collectively, the "Parties").

2. The Software. In this Agreement, "Software" means the program modules and features of the Juniper or Juniper-supplied software, and updates and releases of such software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller. "Embedded Software" means Software which Juniper has embedded in the Juniper equipment.

3. License Grant. Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

- a. Customer shall use the Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.
- b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius software on multiple computers requires multiple licenses, regardless of whether such computers are physically contained on a single chassis.
- c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer's use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface,

processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.

- d. For any trial copy of the Software, Customer's right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.
- e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer's enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. Use Prohibitions. Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any 'locked' or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use the Embedded Software on non-Juniper equipment; (j) use the Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.

5. Audit. Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. Confidentiality. The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.

7. Ownership. Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. Warranty, Limitation of Liability, Disclaimer of Warranty. The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.

9. Termination. Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. Taxes. All license fees for the Software are exclusive of taxes, withholdings, duties, or levies (collectively "Taxes"). Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software.

11. Export. Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.

12. Commercial Computer Software. The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14(ALT III) as applicable.

13. Interface Information. To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.

14. Third Party Software. Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at <http://www.gnu.org/licenses/gpl.html>, and a copy of the LGPL at <http://www.gnu.org/licenses/lgpl.html>.

15. Miscellaneous. This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient rédigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).

Table of Contents

About This Guide	xvii
Objectives	xvii
Audience	xvii
Documentation Conventions.....	xviii
Related Juniper Networks Documentation.....	xix
Obtaining Documentation.....	xxi
Documentation Feedback	xxii
Requesting Support.....	xxii

Part 1

Operating the SAE

Chapter 1	Overview of the SAE	3
	Role of the SAE	3
	Connections to Managed Devices.....	4
	COPS Connection Between JUNOS Routers and the SAE.....	4
	Beep Connection Between JUNOS Routing Platforms and the SAE	4
	COPS Connection Between CMTS Devices and the SAE	5
	COPS Connection Between Juniper Policy Servers and the SAE.....	5
	SAE Plug-Ins	6
	Internal Plug-Ins	6
	External Plug-Ins	7
	Hosted Plug-Ins	7
	Tracking and Controlling Subscriber and Service Sessions with SAE APIs	8
	SAE Core API.....	8
	SAE CORBA Remote API.....	9
	SAE Accounting	10
	Accounting Policy	11
	Subscription Process.....	11
	Tracking Subscriber Sessions.....	12
	Accounting Plug-Ins	12
	Interim Accounting.....	12
Chapter 2	Configuring the SAE with the SRC CLI	13
	Configuring LDAP Access to Directory Data	13
	Configuring Access to Subscriber Data	14
	Related Information	16
	Configuring Access to Service Data	16
	Related Information	17

Configuring Access to Policy Data	17
Related Information	18
Configuring Access to the Persistent Login Cache	18
Related Information	19
Configuring the Location of Network Device Data	20
Related Information	20
Enabling Automatic Discovery of Changes in SAE Configuration Data	20
Related Information	21
Setting the Timeout and Number of Events for SAE Directory Eventing ..	21
Related Information	21
Storing Subscriber and Service Session Data	22
Session Store Files	22
Active and Passive Session Stores	22
Session Store File Rotation	23
Configuring the Session Store Feature	23
Configuring Session Store Parameters for a Device Driver	23
Related Information	26
Configuring Global Session Store Parameters	26
Related Information	27
Reducing the Size of Objects for the Session Store Feature	27
Related Information	28
Configuring the Number of Threads for Sessions	28

Chapter 3 Configuring the SAE with SDX Configuration Editor 29

Overview of Configuring the SAE	29
Configuring LDAP Access to Directory Data	29
Configuring Access to Subscriber Data	30
User Data Fields.....	31
Configuring Access to Service Data	33
Service Data Fields.....	33
Configuring Access to Policy Data	35
Policy Data Fields.....	36
Configuring Access to the Persistent Login Cache	37
Persistent Login Cache Data Fields.....	37
Configuring the Location of Router, Persistent Login, and Persistent Session Data	39
Router Data, DHCP Persistent Login Information, and Persistent Session Data Fields.....	39
Enabling Automatic Discovery of Changes in SAE Configuration Data	40
Enable Configuration Directory Eventing Field.....	41
Storing Subscriber and Service Session Data.....	41
Session Store Files	41
Active and Passive Session Stores.....	42
Session Store File Rotation	42
Configuring the Session Store Feature.....	42
Configuring Session Store Parameters for a Device Driver.....	43
Session Store Fields.....	44
Configuring Global Session Store Parameters	47
Global Session Store Fields.....	48
Reducing the Size of Objects for the Session Store Feature.....	48
Compressed Serialized Data Field	49
Configuring the Number of Threads for Sessions	50
Number of Threads for Sessions Field	50

Chapter 4 Managing SAE Data with the SRC CLI 51

Commands to Manage SAE	51
Reloading the SAE Data	52
Reloading the SAE Configuration	52
Reloading Services.....	52
Reloading Subscriptions	53
Reloading Interface Classification Scripts.....	53
Reloading Domain Maps.....	53
Updating Memory Usage.....	53
Removing the Directory Blacklist	53
Removing Login Registrations.....	54
Removing Equipment Registrations	54
Modifying Failover Server Parameters.....	55
Shutting Down the Device Drivers	56

Part 2 Using Juniper Networks Routers in the SRC Network

Chapter 5 Using JUNOSe Routers in the SRC Network with the SRC CLI 59

COPS Connection Between JUNOSe Routers and the SAE	60
Adding JUNOSe Routers and Virtual Routers with the CLI	60
Adding Operative JUNOSe Routers and Virtual Routers	60
Adding Routers Individually.....	61
Adding Virtual Routers Individually	62
Related Information.....	63
Configuring the SAE to Manage JUNOSe Routers with the CLI	64
Related Information.....	66
Using SNMP to Retrieve Information from JUNOSe Routers	66
Configuring the SNMP Server on the JUNOSe Router.....	66
Configuring Global SNMP Communities in the SRC Software.....	67
Developing Router Initialization Scripts.....	68
Interface Object Fields.....	68
Required Methods	70
Example: Router Initialization Script.....	70
Specifying Router Initialization Scripts on the SAE with the CLI	70
Accessing the Router CLI	72
Starting the SRC Client on a JUNOSe Router	72
Stopping the SRC Client on a JUNOSe Router	73
Monitoring Interactions Between the SAE and the JUNOSe Router.....	73
Troubleshooting Problems with Managing JUNOSe Routers	74
Troubleshooting the SRC Client on JUNOSe Routers	74
Viewing the State of JUNOSe Device Drivers with the SRC CLI	75
Viewing Statistics for Specific JUNOSe Device Drivers with the SRC CLI ..	76
Viewing Statistics for All JUNOSe Device Drivers with the SRC CLI	76
Viewing the State of JUNOSe Device Drivers with the C-Web Interface....	77
Viewing Statistics for Specific JUNOSe Device Drivers with the C-Web Interface	78
Viewing Statistics for All JUNOSe Device Drivers with the C-Web Interface	79

Chapter 6 Using JUNOSe Routers in the SRC Network with a Solaris Platform 81

COPS Connection Between JUNOSe Routers and the SAE	82
Adding JUNOSe Routers and Virtual Routers	82
Adding Operative JUNOSe Routers and Virtual Routers	82
Adding Routers Individually	83
Router Fields	84
Adding Virtual Routers Individually	85
Virtual Router Fields	86
Specifying the SAEs That Can Manage the Router	89
Adding an SAE	89
Modifying an SAE Address	89
Deleting an SAE Address	90
Configuring the SAE to Manage JUNOSe Routers	90
JUNOSe Router Driver Fields	91
Using SNMP to Retrieve Information from JUNOSe Routers	93
Configuring the SNMP Server on the JUNOSe Router	93
Configuring Global SNMP Communities in the SRC Software	94
Global SNMP Community Fields	95
Developing Router Initialization Scripts	95
Interface Object Fields	96
Required Methods	97
Example: Router Initialization Script	97
Specifying Router Initialization Scripts on the SAE	98
JUNOSe Router Script Fields	98
Updating Local IP Address Pools for JUNOSe VRs	99
Updating Local IP Address Pools with SDX Admin	100
Updating Local IP Address Pools with the poolRepublish Command	100
Syntax of poolRepublish Command	101
Troubleshooting the poolRepublish Command	102
Accessing the Router CLI	103
Using Policy Editor	103
Using SDX Admin	103
Remote Access Fields	104
Starting the SRC Client on a JUNOSe Router	105
Stopping the SRC Client on a JUNOSe Router	106
Monitoring Interactions Between the SAE and the JUNOSe Router	106
Troubleshooting the SRC Client on JUNOSe Routers	107

Chapter 7 Using JUNOS Routing Platforms in the SRC Network with the SRC CLI 109

BEEP Connection Between JUNOS Routing Platforms and the SAE	110
Adding JUNOS Routing Platforms and Virtual Routers	110
Adding Operative JUNOS Routing Platforms	111
Adding Routers Individually	111
Adding Virtual Routers Individually	112
Related Information	113
Configuring the SAE to Manage JUNOS Routing Platforms	113
Related Information	115
Configuring Secure Connections Between the SAE and JUNOS Routing Platforms	115
Manually Obtaining Digital Certificates	116
Obtaining Digital Certificates through SCEP	118
Installing the Server Certificate on the Router	119

Creating a Client Certificate for the Router	120
Installing the Client Certificate on the Router.....	120
Configuring the SAE to Use TLS	120
Configuring TLS on the SAE.....	120
Checking Changes to the JUNOS Configuration	121
Setting Up Periodic Configuration Checking	122
Using SNMP to Retrieve Information from JUNOS Routing Platforms.....	122
Configuring Global SNMP Communities in the SRC Software.....	123
Developing Router Initialization Scripts.....	123
Interface Object Fields.....	123
Required Methods	125
Example: Router Initialization Script.....	125
Specifying Router Initialization Scripts on the SAE	125
Accessing the Router CLI.....	126
Configuring JUNOS Routing Platforms to Interact with the SAE.....	126
Configuring the JUNOS Routing Platform to Apply Changes It Receives from the SAE.....	127
Disabling Interactions Between the SAE and JUNOS Routing Platforms	128
Monitoring Interactions Between the SAE and JUNOS Routing Platforms.....	128
Troubleshooting Problems with the SRC Software Process.....	129
Deleting All SRC Data on JUNOS Routing Platforms.....	129
Viewing the State of JUNOS Device Drivers with the SRC CLI	130
Viewing Statistics for Specific JUNOS Device Drivers with the SRC CLI..	130
Viewing Statistics for All JUNOS Device Drivers with the SRC CLI.....	131
Viewing the State of JUNOS Device Drivers with the C-Web Interface ...	132
Viewing Statistics for Specific JUNOS Device Drivers with the C-Web Interface	133
Viewing Statistics for All JUNOS Device Drivers with the C-Web Interface	134

Chapter 8	Using JUNOS Routing Platforms in the SRC Network with a Solaris Platform	135
<hr/>		
BEEP Connection Between JUNOS Routing Platforms and the SAE	136	
Adding JUNOS Routing Platforms and Virtual Routers.....	136	
Adding Operative JUNOS Routing Platforms	137	
Adding Routers Individually	137	
Router Fields	138	
Adding Virtual Routers Individually	139	
Virtual Router Fields	140	
Specifying the SAEs That Can Manage the Router.....	143	
Adding an SAE	143	
Modifying an SAE Address	144	
Deleting an SAE Address	144	
Configuring the SAE to Manage JUNOS Routing Platforms	144	
JUNOS Router Driver Fields	145	
Configuring Secure Connections Between the SAE and JUNOS Routing Platforms.....	148	
Creating a Server Certificate for the SAE.....	148	
Installing the Server Certificate on the SAE.....	149	
Installing the Server Certificate on the Router.....	150	
Creating a Client Certificate for the Router	150	
Installing the Client Certificate on the Router.....	150	
Installing the Client Certificate on the SAE.....	151	

Configuring the SAE to Use TLS	151
Configuring the Keystore for TLS Certificates and Keys	151
Keystore Fields for the JUNOS Router Driver	152
Checking Changes to the JUNOS Configuration	153
Setting Up Periodic Configuration Checking	154
Configuration Checking Fields for the JUNOS Router Driver	154
Using SNMP to Retrieve Information from JUNOS Routing Platforms.....	156
Configuring Global SNMP Communities in the SRC Software.....	156
Global SNMP Community Fields	156
Developing Router Initialization Scripts.....	157
Interface Object Fields	157
Required Methods	158
Example: Router Initialization Script.....	158
Specifying Router Initialization Scripts on the SAE	159
JUNOS Router Script Fields	159
Accessing the Router CLI.....	160
Using Policy Editor	160
Using SDX Admin	161
Remote Access Fields	161
Configuring JUNOS Routing Platforms to Interact with the SAE.....	162
Configuring the JUNOS Routing Platform to Apply Changes It Receives from the SAE.....	163
Disabling Interactions Between the SAE and JUNOS Routing Platforms	163
Monitoring Interactions Between the SAE and JUNOS Routing Platforms.....	164
Troubleshooting SRC Problems on JUNOS Routing Platforms	164
Troubleshooting Problems with the SRC Software Process	164
Troubleshooting Problems with Interfaces.....	165
Troubleshooting Problems with Services	168
Deleting All SRC Data on JUNOS Routing Platforms.....	171

Part 3

Locating Subscriber Management Information

Chapter 9	Locating Subscriber Information with the NIC	175
Locating Subscriber Management Information.....	175	
NIC Client/Server Mode	176	
NIC Local Host Mode	176	
Mapping Subscribers to a Managing SAE.....	177	
NIC Proxies and NIC Locators.....	177	
NIC Hosts	178	
NIC Agents	178	
NIC Resolvers.....	178	
High Availability for NIC.....	178	
High Availability in Existing NIC Configurations	179	
NIC Replication.....	179	
Planning a NIC Implementation	181	
NIC Configuration Scenarios.....	182	
NIC Agents Used in the NIC Configuration Scenarios	184	
Router Initialization Scripts with NIC Configuration Scenarios.....	186	

Chapter 10	Configuring NIC with the SRC CLI	187
	Configuration Statements for the NIC.....	188
	Configuration Statements for NIC Operating Properties.....	188
	Configuration Statements for NIC Scenarios	189
	Configuration Statements for NIC Logging.....	190
	Before You Configure the NIC	190
	Configuring the NIC from the SRC CLI.....	191
	Starting the NIC from the SRC CLI.....	191
	Reviewing and Changing Operating Properties for NIC	192
	Reviewing the Default NIC Operating Properties	192
	Changing NIC Operating Properties	193
	Configuring NIC Replication	194
	Configuring a NIC Scenario	194
	Configuring Directory Agents.....	197
	Configuring SAE Plug-In Agents	199
	Configuring the SAE to Communicate with SAE Plug-In Agents	
	When You Use NIC Replication	201
	Obtaining Interface Configuration Information for	
	OnePopStaticRouteIp	202
	Verifying Configuration for the NIC	203
	Testing a NIC Resolution	203
	Stopping a NIC Host on a C-series Platform	203
	Restarting the NIC.....	204
	Restarting a NIC Agent.....	204
	Restarting a NIC Resolver.....	204
	Changing NIC Configurations	204
Chapter 11	Configuring NIC on a Solaris Platform	207
	Before You Configure NIC Hosts.....	208
	Configuring NIC Hosts to Resolve Requests.....	208
	Configuring Operating Parameters for NIC Hosts	209
	Directory Connection Properties for NIC Hosts.....	210
	NIC Host Properties	212
	Additional Properties for NIC Hosts	212
	Reviewing Basic Configuration for a NIC Host.....	213
	Modifying Basic Configuration for NIC Agents on a NIC Host.....	214
	Configuring Consolidator Agents	214
	Consolidator Agent Fields in Basic Editing Level	215
	Configuring Directory Agents.....	215
	Directory Agent Fields in Basic Editing Level.....	216
	Configuring SAE Plug-In Agents	217
	SAE Plug-In Agent Fields in Basic Editing Level	218
	Configuring the SAE for SAE Plug-In Agents.....	219
	Configuring the SAE to Communicate with SAE Plug-In Agents	
	When You Use NIC Replication	219
	Configuration Fields for SAE Plug-In Agents	220
	Starting NIC on a Solaris Platform	221
	Stopping a NIC Host on a Solaris Platform	221
	Monitoring NIC Hosts.....	222
	Configuring NIC Replication	222
	Changing NIC Configurations	222

Chapter 12	Obtaining Interface Configuration for OnePopStaticRouteIp	223
	JUNOS Interface Information for OnePopStaticRouteIp	223
	Information Collection for OnePopStaticRouteIp from the Network Publisher	224
	Before You Run the Network Publisher	224
	Configuring the Network Publisher	224
	Network Publisher Configuration File Fields	224
	Logging Configuration Fields	225
	Router Configuration Fields	225
	Filter Configuration Fields	226
	Directory Configuration Fields	227
	Troubleshooting Configuration Fields	228
	Running the Network Publisher	229
	Troubleshooting Router Connections and Configuration for the Network Publisher	229
	Changing the Location of an Input Directory for the Network Publisher	230
	Reviewing the Information Collected from a JUNOS Routing Platform	230
	Reviewing and Editing Interface Information from SDX Admin	231
	NIC Document That Maps Subscriber IP Addresses to a JUNOS Interface	231
Chapter 13	Configuring Applications to Communicate with an SAE	233
	Overview of NIC Proxy Configuration	233
	Before You Configure a NIC Proxy	233
Chapter 14	Using the CLI to Configure SRC Applications to Communicate with an SAE	235
	Configuration Statements for NIC Proxies	235
	Before You Configure a NIC Proxy	236
	Configuring Resolution Information for a NIC Proxy	237
	Changing the Configuration for the NIC Proxy Cache	238
	Configuring a NIC Proxy for NIC Replication	239
	Configuring NIC Test Data from the SRC CLI	241
Chapter 15	Configuring Applications to Communicate with an SAE with SDX Configuration Editor	243
	Configuring an Application to Use a Local NIC Host	243
	Local NIC Host Configuration Fields	244
	Configuring NIC Proxies from SDX Configuration Editor	245
	Configuring Resolution Information for a NIC Proxy	245
	NIC Proxy Resolution Fields	246
	Configuring the NIC Proxy Cache	247
	NIC Proxy Cache Fields	248
	Configuring the NIC Proxy for NIC Replication	249
	NIC Host Selection Fields	249
	Reviewing and Updating the ORB Configuration for Applications That Include a NIC Proxy	251
	Configuring JacORB as the Default ORB	251
	Configuring One Web Application to Use JacORB	252
	Configuring a Web Application Server to Use JacORB	253

Testing Applications by Using a NIC Proxy Stub	253
Configuring a NIC Proxy Stub from SDX Configuration Editor	254
NIC Proxy Stub Fields	255
Configuring a NIC Proxy Stub from SDX Admin	255
Configuring the Test Data	256
Configuring a NIC Proxy Stub to Use a corbaloc URL to Test Data...	256
Configuring a NIC Proxy Stub to Use a File URL to Test Data	257
Configuring a NIC Proxy Stub to Use an IOR to Test Data	257
Monitoring NIC Proxies	258
Chapter 16 Developing Applications That Use NIC	261
External Application Requirements for NIC	261
External Non-Java Applications That Use NIC	261
Creating a NIC Locator to Include with a Non-Java Application	262
External Java Applications That Use NIC	263
Developing a Java Application to Communicate with a NIC Proxy	263
Instantiating a Configuration Manager	264
Passing a Reference to the Configuration Manager to the	
NIC Factory	264
Instantiating the NIC Factory Class	264
Initializing Logging	265
Instantiating the NIC Proxy	266
Managing a Resolution Request	266
Deleting Invalid Results from the NIC Proxy's Cache	268
Removing the NIC Proxies	268
Updating Information About Address Pools	269
Chapter 17 Configuring NIC Host Redundancy	271
Overview of NIC Host Redundancy	271
Before You Configure NIC Host Redundancy	272
Configuring NIC Host Redundancy	272
Configuring Redundant Hosts	273
Redundant Hosts Fields	273
Configuring Communities	274
Communities Fields	275
Configuring Monitors	275
Configuring Operating Parameters for Monitors	276
JRE Properties for a Redundant Monitor	277
Configuring a Monitor Process	277
Monitor Field	278
Configuring JacORB Properties on Redundant NIC Hosts	278
JacORB Properties for Timeouts	279
Starting NIC Monitors	279
Starting NIC Hosts	279
Verifying That a Monitor Is Running	280
Stopping a Monitor	280
Optimizing Performance of the NIC Proxy for NIC Host Redundancy	280
JacORB Properties for Retry Intervals	281
Viewing Log Files for NIC Hosts and Monitors from the Solaris CLI	282
Clearing Persistent Data and Logs for NIC Hosts and NIC Monitors	282

Chapter 18	NIC Resolution Process	283
	Overview of the Resolution Process	283
	NIC Realms.....	284
	Key to Value Resolution.....	284
	NIC Data Types	285
	Constraints as NIC Data Types	287
Chapter 19	Customizing a NIC Configuration	289
	Before You Customize a NIC Configuration	289
	Planning a Custom NIC Configuration	290
	Creating a Custom NIC Configuration by Adding Components to an Existing Scenario	290
	Creating a Custom NIC Configuration by Removing Components in an Existing Scenario	292
	Qualifying NIC Data Types	293
	Qualifying a NIC Data Type from SDX Admin.....	294
	Qualifying a NIC Data Type from SDX Configuration Editor	294
	Managing Directory Changes for the Directory Agent	295
Chapter 20	Reviewing the NIC Configuration	297
	Reviewing the Configuration for NIC Realms	298
	NIC Resolution Transition Fields.....	299
	NIC Resolvers Fields	300
	Consolidator Agents	300
	Reviewing the Configuration of Consolidator Agents	301
	Consolidator Agent Fields.....	301
	Directory Agents	303
	Reviewing the Configuration of Directory Agents	303
	Directory Agent Fields.....	304
	Directory Eventing Fields	307
	SAE Plug-In Agents.....	307
	Reviewing the Configuration of SAE Plug-In Agents.....	309
	SAE Plug-In Agent Fields	309
	Properties Agents.....	313
	Configuring Properties Agents	314
	Properties Agent Fields	314
	XML Agents.....	316
	Configuring XML Agents	317
	XML Agent Fields	317
	Reviewing and Changing the Configuration for a NIC Host Instance	321
	NIC Host Fields	322
	Configuring Logging for NIC	322
	Reviewing the Configuration for NIC Locators.....	323
	Managing NIC Resolvers	324
Chapter 21	NIC Configuration Scenarios	327
	Overview of NIC Configuration Scenarios	328
	OnePop Scenario	328
	Centralized Configuration	329
	Distributed Configuration	330
	Redundancy	331

OnePopPcmm Scenario	332
Centralized Configuration	332
Distributed Configuration	333
OnePopDynamicIp Scenario	334
Centralized Configuration	334
Distributed Configuration	335
OnePopSharedIp Scenario	336
Centralized Configuration	336
Distributed Configuration	337
OnePopStaticRouteIp	338
Centralized Configuration	339
Distributed Configuration	340
OnePopAcctId Scenario	340
OnePopLogin Scenario	342
Centralized Configuration	343
Distributed Configuration	345
OnePopPrimaryUser	345
Centralized Configuration	346
Distributed Configuration	346
OnePopDnSharedIp Scenario	347
Centralized Configuration	348
Distributed Configuration	349
OnePopAllRealms Scenario	351
Centralized Configuration	351
Distributed Configuration	352
MultiPop Scenario	355
IP Realm	356
Shared IP Realm	357
DN Realm	359

Index	361
--------------	------------

About This Guide

This preface provides the following guidelines for using the *SRC-PE Software Network Guide: SAE, Juniper Networks Routers, and NIC*.

- Objectives on page xvii
- Audience on page xvii
- Documentation Conventions on page xviii
- Related Juniper Networks Documentation on page xix
- Obtaining Documentation on page xxi
- Documentation Feedback on page xxii
- Requesting Support on page xxii

Objectives

This guide provides an overview of the service activation engine (SAE) and describes how to use it. The guide also describes how to use Juniper Networks routers in the Session and Resource Control (SRC) network, and how to use the NIC to locate subscriber management information.



NOTE: If the information in the latest *SRC Release Notes* differs from the information in this guide, follow the *SRC Release Notes*.

Audience

This guide is intended for experienced system and network specialists working with JUNOS routers and JUNOS routing platforms in an Internet access environment. We assume that readers know how to use the routing platforms, directories, and RADIUS servers that they will deploy in their SRC networks. For users who deploy the SRC software on a Solaris platform, we also assume that readers are familiar with the Lightweight Directory Access Protocol (LDAP) and the UNIX operating system.

If you are using the SRC software in a cable network environment, we assume that you are familiar with the *PacketCable Multimedia Specification* (PCMM) as defined by Cable Television Laboratories, Inc. (CableLabs) and with the Data-over-Cable Service Interface Specifications (DOCSIS) 1.1 protocol. We also assume that you are familiar with operating a multiple service operator (MSO) multimedia-managed IP network.

Documentation Conventions

The sample screens used throughout this guide are representations of the screens that are displayed when you install and configure the SRC software. The actual screens may differ.

For convenience and clarity, the installation and configuration examples show default file paths. If you do not accept the installation defaults, your paths will vary from the examples.

Table 1 defines notice icons used in this guide. Table 2 defines text conventions used throughout the documentation.

Table 1: Notice Icons




Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury.

Table 2: Text Conventions

Convention	Description	Examples
Bold typeface	<ul style="list-style-type: none"> ■ Represents keywords, scripts, and tools in text. ■ Represents a GUI element that the user selects, clicks, checks, or clears. 	<ul style="list-style-type: none"> ■ Specify the keyword exp-msg. ■ Run the install.sh script. ■ Use the pkgadd tool. ■ To cancel the configuration, click Cancel.
Bold sans serif typeface	Represents text that the user must type.	<code>user@host# set cache-entry-age cache-entry-age</code>
Monospace sans serif typeface	Represents information as displayed on your terminal's screen, such as CLI commands in output displays.	<pre> nic-locators { login { resolution { resolver-name /realms/login/A1; key-type LoginName; value-type SaeId; } } } </pre>

Table 2: Text Conventions (continued)

Convention	Description	Examples
Regular sans serif typeface	<ul style="list-style-type: none"> ■ Represents configuration statements. ■ Indicates SRC CLI commands and options in text. ■ Represents examples in procedures. ■ Represents URLs. 	<ul style="list-style-type: none"> ■ <code>system ldap server { stand-alone;</code> ■ Use the <code>request sae modify device failover</code> command with the <code>force</code> option. ■ <code>user@host# . . .</code> ■ <code>http://www.juniper.net/techpubs/software/management/sdx/api-index.html</code>
<i>Italic sans serif typeface</i>	Represents variables in SRC CLI commands.	<code>user@host# set local-address local-address</code>
Angle brackets	In text descriptions, indicate optional keywords or variables.	Another runtime variable is <code><gfwif></code> .
Key name	Indicates the name of a key on the keyboard.	Press Enter.
Key names linked with a plus sign (+) .	Indicates that you must press two or more keys simultaneously.	Press Ctrl + b.
<i>Italic typeface</i>	<ul style="list-style-type: none"> ■ Emphasizes words. ■ Identifies chapter, appendix, and book names. ■ Identifies distinguished names. ■ Identifies files, directories, and paths in text but not in command examples. 	<ul style="list-style-type: none"> ■ There are two levels of access: <i>user</i> and <i>privileged</i>. ■ <i>Chapter 2, Services</i>. ■ <i>o = Users, o = UMC</i> ■ The <i>/etc/default.properties</i> file.
Backslash	At the end of a line, indicates that the text wraps to the next line.	<code>Plugin.radiusAcct-1.class = \net.juniper.smgmt.sae.plugin\RadiusTrackingPluginEvent</code>
Words separated by the symbol	Represent a choice to select one keyword or variable to the left or right of this symbol. (The keyword or variable may be either optional or required.)	<code>diagnostic line</code>

Related Juniper Networks Documentation

With each SRC software release, we provide the *SRC Documentation CD*, which contains the documentation described in Table 3.

With each SRC Application Library release, we provide the *SRC Application Library CD*. This CD contains both the software applications and the *SRC Application Library Guide*.

The C-Web interface, which is based on the J-Web interface, is available for monitoring C-series platforms and the SRC software. For general information about the J-Web interface, see the *J-Web Interface User Guide*.

A complete list of abbreviations used in this document set, along with their spelled-out terms, is provided in the *SRC Getting Started Guide*.

Table 3: Juniper Networks C-series and SRC Technical Publications

Document	Description
Core Documentation Set	
<i>C-series Hardware Guide</i>	Describes the hardware platforms and how to install, maintain, replace, and troubleshoot them. The guide also includes specifications.
<i>SRC-PE Getting Started Guide</i>	Describes the SRC software and explains how to set up an initial configuration and manage a C-series platform. The guide describes how to set up and start the SRC CLI and C-Web, as well as other SRC configurations. It provides information about setting up an initial SRC configuration on a Solaris platform. The guide also describes how to upgrade the SRC software and how to use the SRC configuration tools. It includes reference material for the SRC documentation.
<i>SRC-PE CLI User Guide</i>	Describes how to use the SRC CLI, configure and monitor the platform with the CLI, and control the CLI environment. The guide also describes how to manage SRC components with the CLI.
<i>SRC-PE Network Guide: SAE, Juniper Networks Routers, and NIC</i>	Describes how to use and configure the SAE and the NIC. This guide also provides detailed information for using JUNOS routers and JUNOS routing platforms in the SRC network.
<i>SRC-PE Integration Guide: Network Devices, Directories, and RADIUS Servers</i>	Describes how to integrate external components—network devices, directories, and RADIUS servers—into the SRC network. The guide provides detailed information about integrating specific models of the external components.
<i>SRC-PE Services and Policies Guide</i>	Describes how to work with services and policies. The guide provides an overview, configuration procedures, and management information. The guide also provides information about the SRC tools for configuring policies.
<i>SRC-PE Subscribers and Subscriptions Guide</i>	Describes how to work with residential and enterprise subscribers and subscriptions. The guide provides an overview, configuration procedures, and management information. This guide also provides information about the sample residential portals and enterprise service portals, including the Enterprise Manager Portal.
<i>SRC-PE Monitoring and Troubleshooting Guide</i>	Describes how to use logging, the SNMP agent, the SRC CLI, and the C-Web interface to monitor and troubleshoot SRC components. This guide also describes the SNMP traps.
<i>SRC-PE Solutions Guide</i>	Provides high-level instructions for SRC implementations. The guide documents the following scenarios: managing QoS services on JUNOS routers; managing subscribers in a wireless roaming environment; providing voice over IP (VoIP) services; integrating the SRC software in a PCMM environment, including the use of the Juniper Policy Server (JPS); mirroring subscriber traffic on JUNOS routers; demonstrating network resource management features in a sample IP television (IPTV) application; and demonstrating the integration of prepaid services in a sample application.
<i>SRC-PE CLI Command Reference, Volume 1</i> <i>SRC-PE CLI Command Reference, Volume 2</i>	Together constitute information about command and statement syntax; descriptions of commands, configuration statements, and options; editing level of statement options; and a history of when a command was added to the documentation.
<i>SRC-PE Comprehensive Index</i>	Provides a complete index of the SRC guides, excluding the <i>C-series Hardware Guide</i> and the <i>SRC CLI Command Reference</i> .
<i>J-Web User Interface Guide</i>	Provides general information about the J-Web interface.

Table 3: Juniper Networks C-series and SRC Technical Publications (continued)

Document	Description
Application Library	
<i>SRC Application Library Guide</i>	Describes how to install and work with applications that you can use to extend the capabilities of the SRC software. The guide documents the following applications: SRC-SG (SOAP Gateway) Web applications, applications to integrate the Juniper Networks Intrusion Detection and Protection (IDP) software into an SRC-managed environment, an application to provide endpoint security by integrating Juniper Networks Instant Virtual Extranet (IVE) Host Checker, a traffic-mirroring Web application, an application to integrate IP address managers with the SAE, an application to provide tracking and QoS control at the application level by integrating the SRC software with the Ellacoya deep packet inspection (DPI) platform, an application to control volume usage, and the SRC-ACP (Admission Control Plug-In) application.
Release Notes	
<i>SRC-PE Release Notes</i> <i>SRC Application Library Release Notes</i>	In the <i>Release Notes</i> , you will find the latest information about features, changes, known problems, resolved problems, supported platforms and network devices (such as Juniper Networks routers and CMTS devices), and third-party software. If the information in the <i>Release Notes</i> differs from the information found in the documentation set, follow the <i>Release Notes</i> . Release notes are included in the corresponding software distribution and are available on the Web.

Obtaining Documentation

To obtain the most current version of all Juniper Networks technical documentation, see the products documentation page on the Juniper Networks Web site at

<http://www.juniper.net/>

To order printed copies of this manual and other Juniper Networks technical documents, or to order a documentation CD, which contains this manual, contact your sales representative.

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation to better meet your needs. Send your comments to techpubs-comments@juniper.net, or fill out the documentation feedback form at

<http://www.juniper.net/techpubs/docbug/docbugreport.html>

If you are using e-mail, be sure to include the following information with your comments:

- Document name
- Document part number
- Page number
- Software release version

Requesting Support

For technical support, open a support case using the Case Manager link at <http://www.juniper.net/support/> or call 1-888-314-JTAC (from the United States, Canada, or Mexico) or 1-408-745-9500 (from elsewhere).

Part 1

Operating the SAE

Chapter 1

Overview of the SAE

This chapter gives an overview of the features of the SAE. Topics include:

- Role of the SAE on page 3
- Connections to Managed Devices on page 4
- SAE Plug-Ins on page 6
- Tracking and Controlling Subscriber and Service Sessions with SAE APIs on page 8
- SAE Accounting on page 10

Role of the SAE

The SAE is the core manager of the SRC network. It interacts with other systems, such as Juniper Networks routers, cable modem termination system (CMTS) devices, directories, Web application servers, and RADIUS servers, to retrieve and disseminate data in the SRC environment. The SAE authorizes, activates and deactivates, and tracks subscriber and service sessions. It also collects accounting information about subscribers and services.

The SAE makes decisions about the deployment of policies on JUNOSe routers and JUNOS routing platforms. When a subscriber's IP interface comes up on the router, the SAE determines whether it manages the interface. If the interface is managed—or controlled—by the SAE, the SAE sends the subscriber's default policy configuration to the router. These default policies define the subscriber's initial network access. When the subscriber activates a value-added service, the SAE translates the service into lists of policies and sends them to the router.

The SAE also provides plug-ins and application programming interfaces (APIs) that extend the capabilities of the SRC software.

Connections to Managed Devices

This section describes the connections between the SAE and Juniper Networks routers, CMTS devices, and the Juniper Policy Server (JPS).

COPS Connection Between JUNOS Routers and the SAE

The SAE and JUNOS routers communicate using the Common Open Policy Service (COPS) protocol. The SAE supports two versions of COPS:

- COPS usage for policy provisioning (COPS-PR)
- COPS External Data Representation Standard (XDR) mode

The version of COPS that you use depends on the version of COPS that your JUNOS router supports. When you set up your JUNOS router to work with the SAE, you enable either COPS-PR mode or COPS XDR mode. There are no configuration differences on the SAE between COPS-PR and COPS XDR.

The following SRC features require the use of COPS-PR:

- Policy sharing on JUNOS routers
- Multiple classify traffic conditions in policy lists

For more information, see one of the following:

- *Chapter 5, Using JUNOS Routers in the SRC Network with the SRC CLI*
- *Chapter 6, Using JUNOS Routers in the SRC Network with a Solaris Platform.*

Beep Connection Between JUNOS Routing Platforms and the SAE

The SAE interacts with a JUNOS software process, referred to as the SRC software process, on a JUNOS routing platform. The SAE and the SRC software process communicate using the Blocks Extensible Exchange Protocol (BEEP).

When a JUNOS routing platform that the SAE manages goes online, it initiates a BEEP session for the SAE. The SAE gets configuration information from the router, and then it builds and installs the policies that control the router's behavior. If the policies are subsequently modified in the directory, the SAE builds a new configuration and reconfigures the interface on the JUNOS routing platform.



NOTE: The SAE manages interfaces on JUNOS routing platforms only when the interfaces are configured in the global configuration and the router sends added, changed, or deleted notifications to the SAE. Router administrators should not manually change the configuration of interfaces that the SAE is managing. If you manually change a configuration, you must remove the SAE from the system.

When there are configuration changes on the router, the router sends a notification to the SAE through the BEEP connection. The notification does not include the content of the configuration changes. When the SAE receives the notification, it uses its JUNOScript client to get the changed configuration from the router.

Interfaces that have been deleted from the router along with their associated objects (sessions, policies) remain on the router until state synchronization occurs.

For more information, see one of the following:

- *Chapter 7, Using JUNOS Routing Platforms in the SRC Network with the SRC CLI*
- *Chapter 8, Using JUNOS Routing Platforms in the SRC Network with a Solaris Platform*

COPS Connection Between CMTS Devices and the SAE

The SAE uses the COPS protocol as specified in the PacketCable Multimedia Specification PKT-SP-MM-I03-051221 to manage *PacketCable Multimedia Specification* (PCMM)-compliant CMTS devices in a cable network environment. The SAE connects to the CMTS device by using a COPS over Transmission Control Protocol (TCP) connection.

In cable environments, the SAE manages the connection to the CMTS device. The CMTS device does not provide address requests or notify the SAE of new subscribers, subscriber IP addresses, or any other attributes. IP address detection and all other subscriber attributes are collected outside of the COPS connection to the CMTS device. The SAE uses COPS only to push policies to the CMTS device and to learn about the CMTS status and usage data.

Because the CMTS device does not have the concept of interfaces, the SRC software uses pseudointerfaces to model CMTS subscriber connections similar to subscriber connections for JUNOS routing platforms and JUNOSe routers.

For more information, see *SRC-PE Solutions Guide, Chapter 4, Providing Premium Services in a PCMM Environment*.

COPS Connection Between Juniper Policy Servers and the SAE

When the SAE is acting as an application manager in a PCMM environment, it connects to the JPS through an interface on the JPS. The JPS uses the COPS protocol as specified in the PacketCable Multimedia Specification PKT-SP-MM-I03-051221 for its interface connections. The JPS communicates with the application manager by using a COPS over TCP connection.

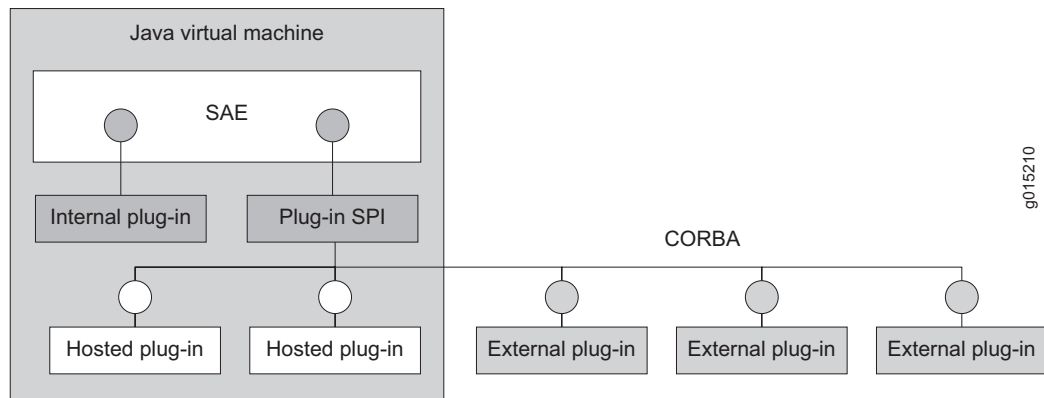
For more information, see *SRC-PE Solutions Guide, Chapter 11, Using PCMM Policy Servers*.

SAE Plug-Ins

Plug-ins are software programs that extend the capabilities of existing programs and make them more flexible. SRC plug-ins provide authentication, authorization, and tracking capabilities.

There are three types of plug-ins: internal, hosted, and external. Internal plug-ins communicate directly with the SAE. Hosted and external plug-ins implement a published Common Object Request Broker Architecture (CORBA)-based service provider interface (SPI), which means that anyone with access to the interface specification can create plug-ins that work with the SRC software. Figure 1 gives an overview of the plug-in architecture.

Figure 1: SAE Plug-In Architecture



Internal Plug-Ins

The SRC software provides internal plug-ins that perform a range of authentication, authorization, and tracking functions. With these plug-ins, you can, for example, authenticate subscribers, authorize subscriptions and sessions, authorize IP address requests from DHCP clients, track subscriber activity and service use, track quality of service (QoS) services and attach and remove QoS profiles as needed, and limit the number of authenticated subscribers who connect to an IP interface on the router.

Internal plug-ins implement an interface that communicates directly with the SAE. They have the following characteristics:

- Run within the SAE's Java Virtual Machine (JVM)
- Are started and stopped with the SAE
- Are implemented in Java

The core SRC software provides a set of internal plug-ins. See *SRC-PE Subscribers and Subscriptions Guide, Chapter 8, Overview of Plug-Ins Included with the SAE*.

External Plug-Ins

The SRC software includes the SAE CORBA plug-in SPI. This SPI allows you to implement external plug-ins in any language that supports CORBA (for example, Java, C + + , Python), which makes it easy to integrate the SAE with operations support system (OSS) software written in a wide variety of languages and distributed across a variety of hardware and operating system platforms.

External plug-ins link a service provider's OSS with the SAE so that the OSS is notified of events in the life cycle of SAE sessions. For example, plug-ins can be notified when a subscriber attempts to log in and begins the authentication and authorization process. This notification makes it possible for the plug-in to consult general data and resource allocation information that is available to the OSS, and use that information to make authorization decisions.

The SPI also sends session-tracking events when sessions start, on an interim basis, and when sessions stop. Plug-ins can set session timeouts as a response to both session start and interim events. This capability enables the development of prepaid applications where the plug-in consults the subscriber's current account balance before it makes the decision to extend or reduce a session timeout.

External plug-ins have the following characteristics:

- Run outside the SAE's JVM, either in the same or in a different server
- Are implemented in any language that supports CORBA
- Communicate with the SAE using CORBA
- Support the admission control or prepaid demo plug-in, which can be purchased separately from the SRC software.

To configure the SAE for external plug-ins, see one of the following:

- *SRC-PE Subscribers and Subscriptions Guide, Chapter 9, Configuring Internal, External, and Synchronization Plug-Ins with the SRC CLI*
- *SRC-PE Subscribers and Subscriptions Guide, Chapter 10, Overview of Configuring Plug-Ins for Solaris Platforms*

The interface definition language (IDL) code and online documentation for the SAE CORBA Plug-In SPI is on the Juniper Networks Web site at <http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

The IDL code is also in the software distribution in the file *SDK/idl/sspPlugin.idl*. Online documentation for the *sspPlugin.idl* file is at */SDK/doc/idl/sspPlugin/html/index.html*.

Hosted Plug-Ins

Hosted plug-ins, like the external ones, implement the CORBA interface. Unlike the external ones, hosted plug-ins are instantiated (that is, hosted) by the SAE. As a result, they live in the same JVM process as the host SAE, which means that hosted plug-ins must be implemented in Java.

Hosted plug-ins have the following characteristics:

- Run within the SAE's JVM
- Communicate with SAE using CORBA
- Are started and stopped with SAE
- Are implemented using a published interface

Tracking and Controlling Subscriber and Service Sessions with SAE APIs

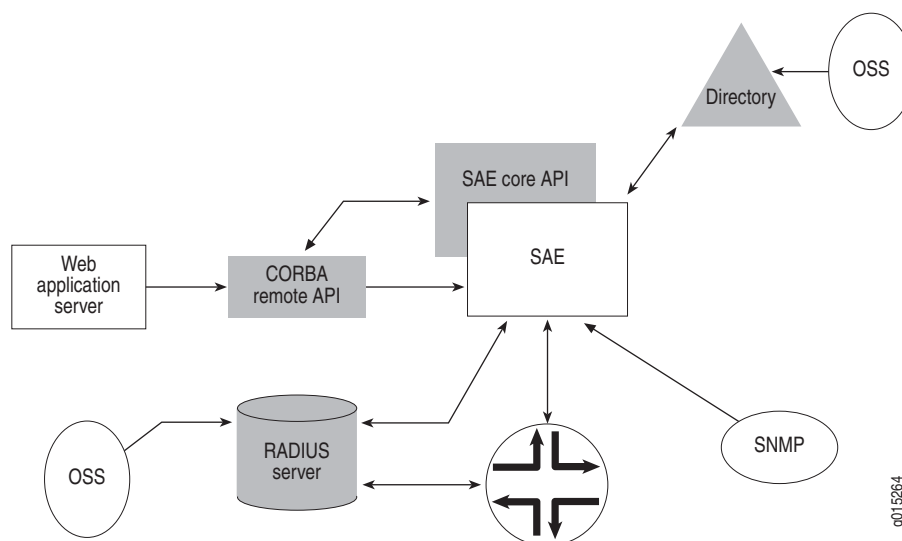
The SAE provides two public APIs:

- SAE core API
- SAE CORBA remote API

Through these interfaces, an external application can track and control subscriber and service sessions.

Figure 2 illustrates the SAE APIs.

Figure 2: SRC SAE APIs



SAE Core API

The SAE core API is used to control the behavior of the SRC software. There are many uses of the SAE core API. For example, it can be used to provide:

- Subscriber credentials (username/password)
- Requests for service activation/deactivation for a subscriber

This API can be used by a Java application running in the same JVM as the SAE. For example, you can access the SAE core API from plug-ins that are hosted by the SAE, or you can use the SAE core API to write your own extensions of the SAE remote interface by using CORBA or the SAE script interface modules.

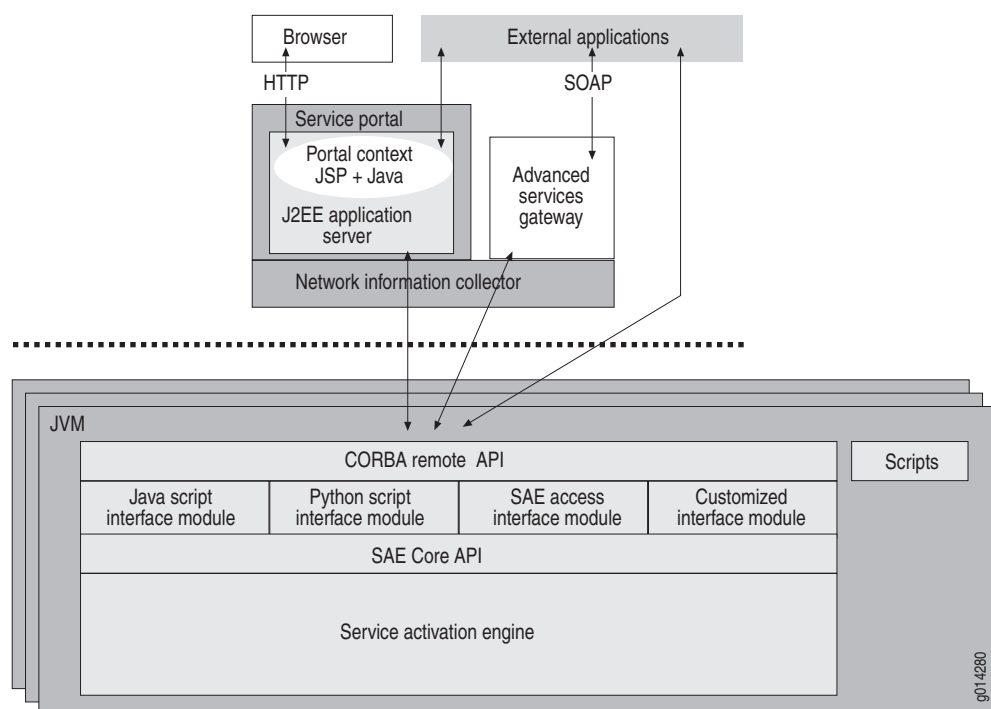
SAE CORBA Remote API

This API provides a way to use external applications with the SRC software (see Figure 3). All functions that are available through the SAE core API are available through the CORBA remote API. The remote API provides several remote interfaces that allow customization of the API for special needs. The remote interface comprises an interface module manager and a set of interface modules. We provide the following interface modules with the SRC software:

- SAE access interface module—Provides remote access to the SAE core API
- Java script interface module—Allows you to control the SAE with a Java script
- Python script interface module—Allows you to control the SAE with a Python script
- Event notification interface module—Allows you to integrate the SAE with external IP address managers

You can also create custom interface modules that allow external applications to extend the capabilities of the SAE. To do so, you must define the interface module in CORBA IDL and implement it in Java.

The remote interface publishes one object reference that acts as the interface module manager. External applications communicate through CORBA with the interface module manager to retrieve a particular interface module. That interface module runs in the same JVM as the SAE and has full access to the SAE core API.

Figure 3: Remote Interface on the SAE

For more information about the SAE CORBA remote API, including the interfaces, properties, and methods, see the online documentation on the Juniper Networks Web site at

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html> or in the SRC software distribution in *SDK/doc/idl/sae/html/index.html*.

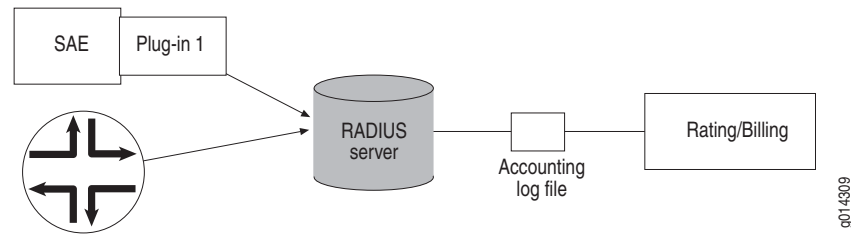
SAE Accounting

The router and the SAE generate RADIUS accounting records when subscribers access the Internet and use value-added services. The records are sent to RADIUS accounting servers and are logged in accounting log files, or they are sent to accounting flat files. External systems collect the accounting log files and feed them to a rating and billing system.

The SRC software allows a variety of accounting deployments. This section shows the standard deployment that we supply, a second option that does not depend on a RADIUS server, and a third option in which customers develop their own deployment by choosing a CORBA plug-in.

In the standard SRC deployment (see Figure 4), the router and the SAE are clients of the RADIUS accounting server. They pass subscriber accounting information to a designated RADIUS accounting server in an accounting request. The RADIUS accounting server receives the accounting request and creates accounting log files.

The SRC software works with other AAA RADIUS servers; however, we validate the SRC software only with Merit, Interlink RAD-Series AAA RADIUS Server, or Juniper Networks Steel-Belted Radius/SPE server.

Figure 4: Sending Accounting Data to a RADIUS Server

A second option, shown in Figure 5, uses an accounting flat file generated directly by the SAE, without a RADIUS server.

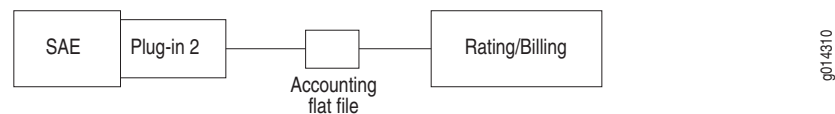
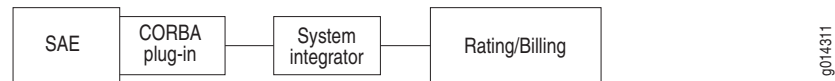
Figure 5: Sending Accounting Data to an Accounting File

Figure 6 illustrates a third possibility, one in which the customer uses a CORBA plug-in of his or her own choice.

Figure 6: Customer Choice for SRC Accounting Deployment

Accounting Policy

The SAE defines the policies that control the network traffic for the subscriber based on the subscriber's subscriptions. It also determines the accounting statistics collected for the subscribed service.

While defining the policies for a service, the SAE can choose the policy rules to be used for accounting per interface direction (ingress and egress). Statistics are collected for the chosen policy rules for the service and are sent to the RADIUS accounting server. The SAE can also decide not to collect any policy rule-specific statistics for the service. In this case, only session times are sent to the accounting system when the service is deactivated. When choosing multiple policy rules on traffic direction for statistics collection, the SAE summarizes the statistics by adding the individual values.

Subscription Process

After an outsourced service has been set up, subscribers can order primary access or value-added services from retailers, who in turn notify the wholesaler of the new end subscription. Conversely, accounting data is collected by the wholesaler and communicated to the retailer to provide enough data for the retailer to bill the subscriber.

The overall subscription process is simplified:

- The subscriber has no need to interact with another party or a device other than the router.
- When the subscriber goes to the Web portal and selects the service, the subscription activation is triggered.
- The subscriber's portal page adjusts to display the new service.
- Accounting data is generated, identifying the service being tracked for the subscriber.

Tracking Subscriber Sessions

The intelligent service accounting function of the SRC software tracks the subscription activity for each subscriber and each service session. It collects usage information and passes the information to the appropriate rating and billing system.

Multiple service sessions can be activated simultaneously for a subscriber and can be tracked separately from an accounting standpoint.

Events are generated when service sessions are activated and deactivated, and during interim accounting updates.

Accounting Plug-Ins

Plug-ins allow service providers to easily extend the capabilities of their systems through the use of plug-in software. See *SAE Plug-Ins* on page 6.

Interim Accounting

The router and SAE generate interim accounting records for broadband primary services (through PPP) and value-added services, respectively. RADIUS servers log the interim records in their accounting log files when interim accounting is enabled.

The external rating system calculates the charges by using interim records instead of stop records for timeout sessions. The calculation occurs when the last record is interim and for open sessions whose last record at the end of a billing cycle is interim.

An accounting interim interval is defined for each service and applied to all subscriptions to that service. The router and SAE generate accounting requests with a status of interim for every period of time specified with the interim value.

The router receives an accounting interim value for a session through a RADIUS server when the router makes an authentication request. If the RADIUS server does not provide a value, then the router does not generate interim accounting records.

The SAE obtains an accounting interim value from the directory. When the accounting interim value is not stored, the SAE uses global values. When a value equals zero, the SAE does not generate interim accounting records.

Chapter 2

Configuring the SAE with the SRC CLI

This chapter describes how to use the SRC CLI to configure general SAE properties. You can use the SRC CLI to configure the SAE on a Solaris platform or on a C-series platform.

You can also use SDX Configuration Editor to configure the SAE on Solaris platforms. See *Chapter 3, Configuring the SAE with SDX Configuration Editor*.

Topics in this chapter include:

- Configuring LDAP Access to Directory Data on page 13
- Storing Subscriber and Service Session Data on page 22
- Configuring the Session Store Feature on page 23
- Configuring the Number of Threads for Sessions on page 28

Configuring LDAP Access to Directory Data

The SRC software stores subscriber, service, persistent login, policy, router, and cached subscriber profiles and session data in a directory. The SAE uses LDAP to store and retrieve the data.

If you do not store data in the local directory, you need to configure the LDAP connections to the directories in which the data is stored. You can also select the filter that the SAE uses to search for subscriptions in the directory and directory eventing parameters for data stored in the directory.

The tasks to configure LDAP access to directory data are:

- Configuring Access to Subscriber Data on page 14
- Configuring Access to Service Data on page 16
- Configuring Access to Policy Data on page 17
- Configuring Access to the Persistent Login Cache on page 18
- Configuring the Location of Network Device Data on page 20

- Enabling Automatic Discovery of Changes in SAE Configuration Data on page 20
- Setting the Timeout and Number of Events for SAE Directory Eventing on page 21

Configuring Access to Subscriber Data

Use the following configuration statements to configure access to subscriber data:

```
shared sae configuration ldap subscriber-data {
    subscription-loading-filter (subscriberRefFilter | objectClassFilter);
    load-subscriber-schedules;
    login-cache-dn login-cache-dn;
    session-cache-dn session-cache-dn;
    server-address server-address;
    dn dn;
    authentication-dn authentication-dn;
    password password;
    directory-eventing;
    polling-interval polling-interval;
    (ldaps);
}
```

To configure SAE access to subscriber data:

1. From configuration mode, access the configuration statement that configures SAE access to subscriber data in the directory. In this sample procedure, the subscriber data is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap subscriber-data
```

2. Select the filter that the SAE uses to search for subscriptions in the directory when the SAE loads a subscription to a subscriber reference filter.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set subscription-loading-filter (subscriberRefFilter | objectClassFilter)
```

3. (Optional) Enable loading of subscriber schedules.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set load-subscriber-schedules
```

4. Specify the subtree in the directory in which subscriber information is stored.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set login-cache-dn login-cache-dn
```

5. Specify the subtree in the directory in which persistent session data is cached.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set session-cache-dn session-cache-dn
```

6. (Optional) Specify the directory server that stores subscriber information.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set server-address server-address
```

7. Specify the subtree in the directory where subscriber data is cached.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set dn dn
```

8. (Optional) Specify the DN that the SAE uses to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set authentication-dn authentication-dn
```

9. (Optional) Specify the password used to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set password password
```

10. (Optional) Enable automatic discovery of changes in subscriber profiles.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set directory-eventing
```

11. Set the frequency for checking the directory for updates.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set polling-interval polling-interval
```

12. Enable LDAPS as the secure protocol for connections to the server that stores subscriber data.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# set ldaps
```

13. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration ldap subscriber-data]
user@host# show
subscription-loading-filter objectClassFilter;
load-subscriber-schedules;
login-cache-dn o=users,<base>;
session-cache-dn o=PersistentSessions,<base>;
server-address 127.0.0.1;
dn o=users,<base>;
authentication-dn cn=ssp,o=components,o=operators,<base>;
password *****;
directory-eventing;
polling-interval 30;
ldaps;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring Access to Service Data

Use the following configuration statements to configure access to service data:

```
shared sae configuration ldap service-data {
    server-address server-address;
    dn dn;
    authentication-dn authentication-dn;
    password password;
    directory-eventing;
    polling-interval polling-interval;
    (ldaps);
}
```

To configure SAE access to service data:

1. From configuration mode, access the configuration statement that configures SAE access to service data in the directory. In this sample procedure, the service data is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap service-data
```

2. (Optional) Specify the directory server that stores service data.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set server-address server-address
```

3. Specify the subtree in the directory where service data is cached.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set dn dn
```

4. (Optional) Specify the DN that the SAE uses to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set authentication-dn authentication-dn
```

5. (Optional) Specify the password used to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set password password
```

6. (Optional) Enable or disable automatic discovery of changes to service data.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set directory-eventing
```

7. Set the frequency for checking the directory for updates.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# set polling-interval polling-interval
```

8. Enable LDAPS as the secure protocol for connections to the server that stores service data.

```
edit shared sae group se-region configuration ldap service-data]
user@host# set ldaps
```

9. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration ldap service-data]
user@host# show
server-address 10.10.45.3;
dn <base>;
authentication-dn <base>;
password *****;
directory-eventing;
polling-interval 30;
ldaps;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring Access to Policy Data

Use the following configuration statements to configure access to policy data:

```
shared sae configuration ldap policy-data {
  policy-dn policy-dn;
  parameter-dn parameter-dn;
  directory-eventing;
  polling-interval polling-interval;
}
```

To configure SAE access to subscriber data:

1. From configuration mode, access the configuration statement that configures SAE access to policy data in the directory. In this sample procedure, the policy data is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap policy-data
```

2. Specify the subtree in the directory in which policy data stored.

```
[edit shared sae group se-region configuration ldap policy-data]
user@host# set policy-dn policy-dn
```

3. Specify the subtree in the directory in which policy parameter data is cached.

```
[edit shared sae group se-region configuration ldap policy-data]
user@host# set parameter-dn parameter-dn
```

4. (Optional) Enable or disable automatic discovery of changes to policy data.

```
[edit shared sae group se-region configuration ldap policy-data]
user@host# set directory-eventing
```

5. Set the frequency for checking the directory for updates.

```
[edit shared sae group se-region configuration ldap policy-data]
user@host# set polling-interval polling-interval
```

6. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration ldap policy-data]
user@host# show
policy-dn o=Policy,<base>;
parameter-dn o=Parameters,<base>;
directory-eventing;
polling-interval 30;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring Access to the Persistent Login Cache

Use the following configuration statements to configure access to persistent login cache data:

```
shared sae configuration ldap persistent-login-cache {
  server-address server-address;
  dn dn;
  authentication-dn authentication-dn;
  password password;
  directory-eventing;
  polling-interval polling-interval;
  (ldaps);
}
```

To configure SAE access to persistent login cache data:

1. From configuration mode, access the configuration statement that configures SAE access to persistent login cache data in the directory. In this sample procedure, the persistent login cache data is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap
persistent-login-cache
```


2. (Optional) Specify the directory server that stores service data.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set server-address server-address
```

3. Specify the subtree in the directory where persistent login cache data is cached.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set dn dn
```

4. (Optional) Specify the DN that the SAE uses to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set authentication-dn authentication-dn
```

5. (Optional) Specify the password used to authenticate access to the directory server.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set password password
```

6. (Optional) Enable automatic discovery of changes to persistent login cache data.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set directory-eventing
```

7. Set the frequency for checking the directory for updates.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set polling-interval polling-interval
```

8. Enable LDAPS as the secure protocol for connections to the server that stores persistent login cache data.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# set ldaps
```

9. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration ldap persistent-login-cache]
user@host# show
dn "o=authCache, <base>";
directory-eventing;
polling-interval 30;
ldaps;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring the Location of Network Device Data

Use the following configuration statement to configure access to network device data:

```
shared sae configuration ldap {
    network-dn network-dn;
}
```

To configure SAE access to network device data:

1. From configuration mode, access the configuration statement that configures SAE access to network device data in the directory. In this sample procedure, the network device data is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap
```

2. Specify the subtree in the directory where network device data is stored.

```
[edit shared sae group se-region configuration ldap]
user@host# set network-dn network-dn
```

3. Verify your configuration.

```
[edit shared sae group se-region configuration ldap]
user@host# show network-dn
network-dn o=Network,<base>;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Enabling Automatic Discovery of Changes in SAE Configuration Data

Use the following configuration statement to enable automatic discovery of changes in SAE configuration data:

```
shared sae configuration ldap {
    enable-directory-eventing;
}
```

To enable automatic discovery of changes in SAE configuration data:

1. From configuration mode, access the configuration statement that enables automatic discovery of changes in SAE configuration data in the directory. In this sample procedure, automatic discovery is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap
```

2. Enable automatic discovery of changes to SAE configuration data.

```
[edit shared sae group se-region configuration ldap]
user@host# enable-directory-eventing
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Setting the Timeout and Number of Events for SAE Directory Eventing

Use the following configuration statements to set the directory eventing timeout and the number of simultaneous events that the SAE can receive from the directory:

```
shared sae configuration ldap directory-eventing {
    timeout timeout;
    dispatcher-pool-size dispatcher-pool-size;
}
```

To configure the directory eventing timeout and the number of simultaneous events that the SAE can receive from the directory:

1. From configuration mode, access the configuration statement that configures SAE directory eventing. In this sample procedure, directory eventing is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration ldap  
directory-eventing
```

2. Specify the maximum time that the directory eventing system waits for the directory to respond.

```
[edit shared sae group se-region configuration ldap directory-eventing]  
user@host# set timeout timeout
```

3. Specify the number of events that the SAE can receive from the directory simultaneously.

```
[edit shared sae group se-region configuration ldap directory-eventing]  
user@host# set dispatcher-pool-size dispatcher-pool-size
```

4. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration ldap directory-eventing]  
user@host# show  
timeout 60;  
dispatcher-pool-size 1000;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Storing Subscriber and Service Session Data

To aid in recovering from an SAE failover, the SAE stores subscriber and service session data in flat files on the SAE host. The SRC component that controls the storage of session data on the SAE is called the session store. The session store queues data and then writes the data to session store files on the SAE host's disk. After the data has been written to disk, it can survive a server reboot.

You can configure how the SAE stores session data for JUNOSe routers, JUNOS routing platforms, simulated routers, and *PacketCable Multimedia Specification* (PCMM) devices.

Session Store Files

Session store files are numbered flat files. Session store files are located in a directory on the SAE host. You can configure the size of session store files. After the maximum size has been reached, the session store creates a new file and begins writing data to the new file.

Store operations, such as adding a session to the store (put store operations) or removing a session from the store (remove store operations), are queued in a buffer before they are written to the session store file. You can configure parameters that determine when the session store writes a queue to a session store file.

Session store files are deleted if they have not been modified and if no session activity has taken place for one week. All the data files that contain the sessions associated with a particular virtual router are deleted at the same time.

Active and Passive Session Stores

You can have a community of SAEs and duplicate session store data on each SAE in the community in case of an SAE failover. SAE communities are made up of SAEs that you configure as connected SAEs for a virtual router object.

SAEs in a community are given the role of either active SAE or passive SAE. The active SAE keeps session data up to date within the community. Each active session store opens a Transmission Control Protocol (TCP) connection to its passive SAE. The TCP connection triggers the creation of a passive session store in that SAE. When the active session store writes operations to the session store file, it passes them to passive session stores on all SAEs in the community.

When you modify a community, wait for passive session stores on the new community members to be updated before you shut down the currently active SAE. Otherwise, if you add a new member to a community, and then a failover from the current active SAE to the new member is triggered immediately, the new member's session store may not have received all data from the active SAE's session store.

Session Store File Rotation

The session store periodically rotates the session store files. During rotation, the session store copies put store operations for live sessions from the oldest file to the end of the newest file. (Live sessions are sessions that have been created but not yet deleted.) It then deletes the oldest file. Sessions are rotated in batches, and you can configure the number of sessions that are rotated at the same time, and how much disk space is used by live sessions before files are rotated. No session store activity can take place while a batch of sessions is rotated.

Configuring the Session Store Feature

You can configure three things for the session store feature:

- Configure session store parameters for a router or device driver. See *Configuring Session Store Parameters for a Device Driver* on page 23.
- Configure global session store parameters that are shared by all session store instances (active or passive) on the SAE. See *Configuring Global Session Store Parameters* on page 26.
- Reduce the size of session objects that the SAE sends across the network for the session store feature. See *Reducing the Size of Objects for the Session Store Feature* on page 27.

Configuring Session Store Parameters for a Device Driver

Use the following configuration statements to configure session store parameters within a device driver configuration:

```
shared sae configuration driver ( junos | junose | pcmm | simulated | third-party )
session-store {
    maximum-queue-age maximum-queue-age;
    maximum-queued-operations maximum-queued-operations;
    maximum-queue-size maximum-queue-size;
    maximum-file-size maximum-file-size;
    minimum-disk-space-usage minimum-disk-space-usage;
    rotation-batch-size rotation-batch-size;
    maximum-session-size maximum-session-size;
    disk-load-buffer-size disk-load-buffer-size;
    network-buffer-size network-buffer-size;
    retry-interval retry-interval;
    communications-timeout communications-timeout;
    load-timeout load-timeout;
    idle-timeout idle-timeout;
    maximum-backlog-ratio maximum-backlog-ratio;
    minimum-backlog minimum-backlog;
}
```

To configure session store parameters within a device driver configuration:

1. From configuration mode, access the configuration statement that configures the session store for your device driver. In this sample procedure, the session store for a JUNOS device driver is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration driver junos session-store
```

2. (Optional) Specify the maximum age that a queue of buffered store operations (such as adding a session to the store or removing a session from the store) can reach before the queue is written to a session store file.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-queue-age maximum-queue-age
```

3. (Optional) Specify the number of buffered store operations that are queued before the queue is written to a session store file.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-queued-operations maximum-queued-operations
```

4. (Optional) Specify the maximum size that a queue of buffered store operations can reach before the queue is written to a session store file.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-queue-size maximum-queue-size
```

5. (Optional) Specify the maximum size of session store files.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-file-size maximum-file-size
```

6. (Optional) Specify the percentage of space in all session store files that is used by live sessions.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set minimum-disk-space-usage minimum-disk-space-usage
```

7. (Optional) Specify the number of sessions that are rotated from the oldest file to the newest file at the same time that the oldest session store file is rotated.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set rotation-batch-size rotation-batch-size
```

8. (Optional) Specify the maximum size of a single subscriber or service session.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-session-size maximum-session-size
```

9. (Optional) Specify the size of the buffer that is used to load all of a session store's files from disk at startup.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set disk-load-buffer-size disk-load-buffer-size
```

10. (Optional) Specify the size of the buffer that holds messages or message segments that are waiting to be sent to passive session stores.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set network-buffer-size network-buffer-size
```

11. (Optional) Specify the time interval between attempts by the active session store to connect to missing passive session stores.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set retry-interval retry-interval
```

12. (Optional) Specify the amount of time that a session store waits before closing when it is blocked from reading or writing a message.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set communications-timeout communications-timeout
```

13. (Optional) Specify the time that an active session store waits for a passive session store or a passive session store waits for an active session store to load its data from disk before it closes the connection to the session store.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set load-timeout load-timeout
```

14. (Optional) Specify the time that a passive session store waits for activity from the active session store before it closes the connection to the active session store.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set idle-timeout idle-timeout
```

15. (Optional) Specify when the active session store closes the connection to a passive session store because of a backlog of messages waiting to be sent.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set maximum-backlog-ratio maximum-backlog-ratio
```

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# set minimum-backlog minimum-backlog
```

16. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration driver junos session-store]
user@host# show
maximum-queue-age 5000;
maximum-queued-operations 50;
maximum-queue-size 51050;
maximum-file-size 25000000;
minimum-disk-space-usage 25;
rotation-batch-size 50;
maximum-session-size 10000;
disk-load-buffer-size 1000000;
network-buffer-size 51050;
retry-interval 5000;
```

```
communications-timeout 60000;
load-timeout 420000;
idle-timeout 3600000;
maximum-backlog-ratio 1.5;
minimum-backlog 5000000;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring Global Session Store Parameters

This section describes how to configure global session store parameters that are shared by all session store instances (active or passive) on the SAE. You can also configure session store parameters within a device driver configuration. See *Configuring Session Store Parameters for a Device Driver* on page 23.

Use the following configuration statements to configure global session store parameters.

```
shared sae configuration driver session-store {
    ip-address ip-address;
    port port;
    root-directory root-directory;
}
```

To configure global session store parameters:

1. From configuration mode, access the configuration statement that configures the global session store parameters. In this sample procedure, the global session store is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration driver session-store
```

2. (Optional) Specify the IP address or hostname that the session store infrastructure on this SAE uses to listen for incoming TCP connections from active session stores.

```
[edit shared sae group se-region configuration driver session-store]
user@host# set ip-address ip-address
```

3. (Optional) Specify the TCP port number on which the session store infrastructure on this SAE listens for incoming connections from active session stores.

```
[edit shared sae group se-region configuration driver session-store]
user@host# set port port
```


4. (Optional) Specify the root directory in which the session store creates files.

```
[edit shared sae group se-region configuration driver session-store]
user@host# set root-directory root-directory
```

5. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration driver session-store]
user@host# show
ip-address 10.10.70.0;
port 8820;
root-directory var/sessionStore;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Reducing the Size of Objects for the Session Store Feature

You can use serialized data compression to reduce the size of sessions objects that the SAE sends across the network for the session store feature. Enabling this property reduces the size of objects, but increases the CPU load on the SAE.

Use the following configuration statement to specify whether or not session objects are compressed.

```
shared sae configuration {
    compress-session-data;
}
```

To specify whether or not session objects are compressed:

1. From configuration mode, access the sae configuration. In this sample procedure, data compression is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration
```

2. Enable reducing the size of session objects (subscriber and service sessions) that the SAE sends across the network for the session store feature.

```
[edit shared sae group se-region configuration]
user@host# set compress-session-data
```

3. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration]
user@host# show compress-session-data
compress-session-data;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring the Number of Threads for Sessions

Use the following configuration statement to set the number of threads used for session-related activity.

```
shared sae configuration session-job-manager {
    number-of-threads number-of-threads;
}
```

To configure the number of threads used to handle session-related activity:

1. From configuration mode, access the session job manager configuration. In this sample procedure, the number of threads is configured in the se-region group.

```
user@host# edit shared sae group se-region configuration session-job-manager
```

2. Specify the number of threads used for session-related activity.

```
[edit shared sae group se-region configuration session-job-manager]
user@host# set number-of-threads number-of-threads
```

3. (Optional) Verify your configuration.

```
[edit shared sae group se-region configuration session-job-manager]
user@host# show
number-of-threads 10;
```

Chapter 3

Configuring the SAE with SDX Configuration Editor

This chapter describes how to use SDX Configuration Editor to configure general SAE properties. You can use SDX Configuration Editor on a Solaris platform.

You can also use the SRC CLI to configure and SAE on a Solaris platform or on a C-series platform. See *Chapter 2, Configuring the SAE with the SRC CLI*.

Topics in this chapter include:

- Overview of Configuring the SAE on page 29
- Configuring LDAP Access to Directory Data on page 29
- Storing Subscriber and Service Session Data on page 41
- Configuring the Session Store Feature on page 42
- Configuring the Number of Threads for Sessions on page 50

Overview of Configuring the SAE

The SAE property file contains SAE configuration data that is stored in the directory. You can modify the SAE property file with SDX Configuration Editor, SDX Admin, or a standard text editor. This chapter shows how to configure SAE properties with SDX Configuration Editor. Each field description includes a property name, which is used if you modify the properties with SDX Admin or a text editor. For information about modifying the property file with SDX Admin, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 5, Configuring Subscriber-Related Properties on the SAE on a Solaris Platform*.

Configuring LDAP Access to Directory Data

The SRC software stores subscriber, service, persistent login, policy, router, and cached subscriber profiles and session data in a directory. The SAE uses LDAP to store and retrieve the data. You can configure the LDAP connections to the directories in which this data is stored. You can also select the filter that the SAE uses to search for subscriptions in the directory and directory eventing parameters for data stored in the directory.

The tasks to configure LDAP access to directory data are:

- Configuring Access to Subscriber Data on page 30
- Configuring Access to Service Data on page 33
- Configuring Access to Policy Data on page 35
- Configuring Access to the Persistent Login Cache on page 37
- Configuring the Location of Router, Persistent Login, and Persistent Session Data on page 39
- Enabling Automatic Discovery of Changes in SAE Configuration Data on page 40

Configuring Access to Subscriber Data

To use SDX Configuration Editor to configure the LDAP connection from the SAE to the directory in which subscriber data is stored:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab, and expand the **User Data** section.

User Data	
Server Address	127.0.0.1 Disable
Search Base	o=Users, <base>
Authentication DN	cn=ssp,ou=Components,o=Operators,<base> Disable
Password	*** Show Disable
Enable Directory Eventing	Yes ▼
Directory Polling Interval [s]	30
Secured LDAP protocol	LDAPS ▼ Disable
Filter for loading subscriptions	Subscription Objectclass Filter ▼

3. Edit or accept the default values in the fields.
See *User Data Fields* on page 31.
4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

User Data Fields

In SDX Configuration Editor, you can modify the following fields in the User Data section of the LDAP pane in an SAE configuration file.

Server Address

- Disables or enables and identifies the directory server that stores subscriber information.
- Value—IP address or hostname; use a space to separate addresses for multiple directory servers: 127.153.27.1 192.168.0.1
- Default—Disabled
- Property name—`UserDataSource.repository.ldap.server.address`

Search Base

- Subtree in the directory in which subscriber information is stored. When a subscriber logs in to a residential portal, the SAE searches subscriber profiles by mapping the realm of the login name to a retailer object found below the search base.
- Value—`< DN >`
You can use the special value `< base >` to refer to the globally configured base distinguished name (DN).
- Guidelines—Sensible values include `o = Users`, `o = umc` for multidomain support and `retailerName = Retailer`, `o = Users`, `o = umc` for single domain support.
- Default—`o = Users`, `< base >`
- Property name—`UserDataSource.repository.ldap.server.base.dir`

Authentication DN

- Disables or enables and sets the DN that the SAE uses to authenticate access to the directory server. The specified directory entry must exist and have read access to all attributes. The entry must have write access if subscribers are allowed to customize their subscription profiles.
- Value—`< DN >`
You can use the special value `< base >` to refer to the globally configured base DN.
- Default—Disabled, which means that the value configured for the directory is used
- Property name—`UserDataSource.repository.ldap.server.authDN`

Password

- Disables or enables and sets the password used to authenticate access to the directory server. You must configure the password in the directory to authenticate read-access to the directory.
- Value—Text string or base64 string that matches the value of the `userPassword` attribute of the authentication DN

- Default—Disabled, which means that the value configured for the directory is used
- Property name—`UserDataSource.repository.ldap.server.password`

Enable Directory Eventing

- Enables or disables automatic discovery of changes in subscriber profiles.
- Value
 - Yes—Changes in the subscriber profile or subscriptions take effect automatically while the subscriber is logged in.
 - No—Changes in the subscriber profile or subscriptions do not take effect until the next time the subscriber logs in.
- Default—Yes
- Property name—`UserDataSource.repository.ldap.server.des.enable_eventing`

Directory Polling Interval [s]

- Sets the frequency for checking the directory for updates.
- Value—Number of seconds in the range 15–86400
- Default—30
- Property name—`UserDataSource.repository.ldap.server.des.pollinginterval`

Secured LDAP protocol

- Enables or disables LDAPS as the secure protocol for connections to the server that stores subscriber data.
- Value—Enable or Disable
- Default—Disable
- Property name—`UserDataSource.repository.ldap.server.security.protocol`

Filter for loading subscriptions

- Selects the filter that the SAE uses to search for subscriptions in the directory when the SAE loads a subscription.
- Value—Select one of the following values from the drop-down menu:
 - Subscriber reference filter—The SAE runs a search based on the `subscriberRef` attribute in the `umcServiceProfile` object, which is the base object class of the service profile hierarchy. The `subscriberRef` attribute contains a DN that points to the parent of the subscriber object.
 - Subscription Objectclass filter—The SAE performs a one-level search with the directory entry, which represents the subscriber folder as the base DN. The search filter is `(objectClass = sspServiceProfile)`. This method can be slow if you have a large number of subscription entries within the subscriber folder subtree.
- Default—Subscription Objectclass filter
- Property name—`UserDataSource.repository.ldap.server.loadSubscriptionFilter`

Configuring Access to Service Data

To use SDX Configuration Editor to configure the LDAP connection from the SAE to the directory in which service data is stored:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab, and expand the **Service Data** section.

Service Data	
Server Address	127.0.0.1 Enable
Search Base	<base>
Authentication DN	cn=sssp,ou=components,ou=operators,base Enable
Password	■■■■■ Show Enable
Enable Directory Eventing	Yes ▼
Directory Polling Interval [s]	30
Secured LDAP protocol	LDAPS ▼ Enable

3. Edit or accept the default values in the fields.
See *Service Data Fields* on page 33.
4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Service Data Fields

In SDX Configuration Editor, you can modify the following fields in the Service Data section of the LDAP pane in an SAE configuration file.

Server Address

- Disables or enables and identifies the directory server that stores service data.
- Value—IP address or hostname; use a space to separate addresses for multiple directory servers: 127.153.27.1 192.168.0.1
- Default—Disabled, which means that the value configured for the directory is used
- Property name—ServiceDataSource.repository.ldap.server.address

Search Base

- Subtree in the directory in which service information is stored. The SAE loads service definitions on startup and when service reloading is requested.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default— < base >
- Property name—ServiceDataSource.repository.ldap.server.base.dir

Authentication DN

- Disables or enables and sets the DN that the SAE uses to authenticate access to the directory server. The specified directory entry must exist and have read access to all attributes.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—Disabled, which means that the value configured for the directory is used
- Property name—ServiceDataSource.repository.ldap.server.authDN

Password

- Disables or enables and sets the password used to authenticate access to the directory server. You must configure the password in the directory to authenticate read access to the directory.
- Value—Text string or base64 string
- Default—Disabled, which means that the value configured for the directory is used
- Property name—ServiceDataSource.repository.ldap.server.password

Enable Directory Eventing

- Enables or disables automatic discovery of changes in service definitions.
- Value
 - Yes—Changes in service definitions take effect automatically. If a changed service is in use, all service instances are deactivated and then reactivated with the modified settings. Consequently, service may be affected for subscribers who are logged in at the time of the modification.
 - No—Changes in service definitions do not take effect until the SAE is restarted.
- Default—Yes
- Property name—ServiceDataSource.repository.ldap.server.des.enable_eventing

Directory Polling Interval [s]

- Sets the frequency for checking the directory for updates.
- Value—Number of seconds in the range 15–86400
- Default—30
- Property name—ServiceDataSource.repository.ldap.server.des.pollinginterval

Secured LDAP protocol

- Enables or disables LDAPS as the secure protocol for connections to the server that stores service data.
- Value—Enable or Disable
- Default—Disable
- Property name—ServiceDataSource.repository.ldap.server.security.protocol

Configuring Access to Policy Data

To use SDX Configuration Editor to configure the LDAP connection from the SAE to the directory in which policy data is stored:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab, and expand the **Policy Data** section.

Policy Data	
Policy Search Base	<input type="text" value="o=Policies, <base>"/>
Parameter Search Base	<input type="text" value="o=Parameters, <base>"/>
Enable Directory Eventing	<input type="text" value="Yes"/>
Directory Polling Interval [s]	<input type="text" value="30"/>

3. Edit or accept the default values in the fields.

See *Policy Data Fields* on page 36.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Policy Data Fields

In SDX Configuration Editor, you can modify the following fields in the Policy Data section of the LDAP pane in an SAE configuration file.

Policy Search Base

- Subtree in the directory that stores policy data.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*o = Policies, < base >*
- Property name—PolicyDataSource.repository.ldap.baseDN

Parameter Search Base

- Subtree in the directory that stores policy parameter data.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*o = Parameters, < base >*
- Property name—PolicyDataSource.repository.ldap.parameterBaseDN

Enable Directory Eventing

- Enables or disables automatic discovery of changes in policy definitions and in interface classifiers.
- Value
 - Yes—Changes in policy definitions take effect automatically. If a changed policy is in use, all policy instances are deactivated and then reactivated with the modified settings. Consequently, service may be affected for subscribers who are logged in when the change is made.
 - No—Changes in policy definitions do not take effect until the SAE is restarted.
- Default—Yes
- Property name—net.juniper.smgmt.des.enable_eventing

Directory Polling Interval [s]

- Sets the frequency for checking the directory for updates.
- Value—Number of seconds in the range 15–86400
- Default—30
- Property name—net.juniper.smgmt.des.pollinginterval

Configuring Access to the Persistent Login Cache

To use SDX Configuration Editor to configure the LDAP connection from the SAE to the directory in which persistent login cache data is stored:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab, and expand the **Persistent Login Cache** section.

3. Edit or accept the default values in the fields.
See *Persistent Login Cache Data Fields* on page 37.
4. Select **File** > **Save**.
5. Right-click the configuration file, select **SDX System Configuration** > **Export to LDAP Directory**.

Persistent Login Cache Data Fields

In SDX Configuration Editor, you can modify the following fields in the Persistent Login Cache section of the LDAP pane in an SAE configuration file.

Server Address

- Disables or enables and identifies the directory server that stores persistent login data.
- Value—IP address or hostname; use a space to separate addresses for multiple directory servers: 127.153.27.1 192.168.0.1
- Default—Disabled, which means that the value configured for the directory is used
- Property name—UserCacheDataSource.repository.ldap.server.address

Search Base

- Subtree in the directory that stores persistent login cache data.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*ou-authCache, < base >*
- Property name—UserCacheDataSource.repository.ldap.server.base.dir

Authentication DN

- Disables or enables and sets the DN that the SAE uses to authenticate access to the directory server. The specified directory entry must exist and have read access to all attributes.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—Disabled
- Property name—UserCacheDataSource.repository.ldap.server.authDN

Password

- Disables or enables and sets the password used to authenticate access to the directory server. You must configure the password in the directory to authenticate read access to the directory.
- Value—Text string or base64
- Default—ssp
- Property name—UserCacheDataSource.repository.ldap.server.password

Enable Directory Eventing

- Enables or disables automatic discovery of changes to the persistent login cache.
- Value—Yes or No
- Default—No
- Property name—
UserCacheDataSource.repository.ldap.server.des.enable_eventing

Directory Polling Interval [s]

- Sets the frequency for checking the directory for updates.
- Value—Number of seconds in the range 15–86400
- Default—30
- Property name—
UserCacheDataSource.repository.ldap.server.des.pollinginterval

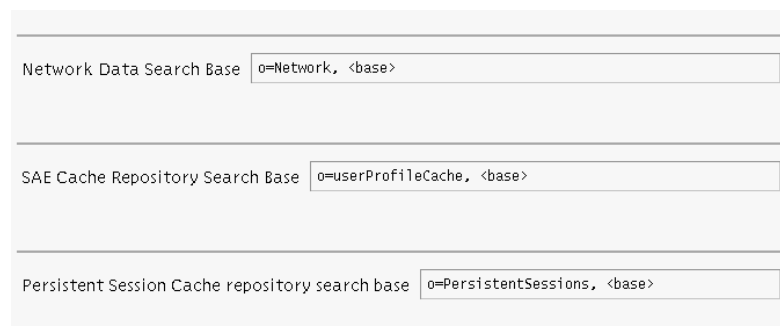
Secured LDAP protocol

- Enables or disables LDAPS as the secure protocol for connections to the server that stores persistent login cache data.
- Value—Enable or Disable
- Default—Disable
- Property name—UserCacheDataSource.repository.ldap.server.security.protocol

Configuring the Location of Router, Persistent Login, and Persistent Session Data

To use SDX Configuration Editor to configure the location of router data, persistent login information for DHCP scenarios, and persistent session data:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab.



The screenshot shows a configuration window with three text input fields, each preceded by a label. The first field is labeled 'Network Data Search Base' and contains the text 'o=Network, <base>'. The second field is labeled 'SAE Cache Repository Search Base' and contains the text 'o=userProfileCache, <base>'. The third field is labeled 'Persistent Session Cache repository search base' and contains the text 'o=PersistentSessions, <base>'. Each field has a small rectangular button to its right.

3. Edit or accept the default values in the Network Data Search Base, SAE Cache Repository Search Base, and Persistent Session Cache Repository Search Base fields.

See *Router Data, DHCP Persistent Login Information, and Persistent Session Data Fields* on page 39.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Router Data, DHCP Persistent Login Information, and Persistent Session Data Fields

In SDX Configuration Editor, you can edit the following fields in the LDAP pane in an SAE configuration file.

Network Data Search Base

- Subtree in the directory that stores router data.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*o = Network, < base >*
- Property name—`NetworkDataSource.repository.ldap.baseDN`

SAE Cache Repository Search Base

- Base DN for storing and retrieving subscriber profiles. This is the directory subtree in which persistent login information is stored for DHCP scenarios.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*o = userProfileCache, < base >*
- Property name—`UserDataSource.repository.ldap.server.cache.dir`

Persistent Session Cache Repository Search Base

- Base DN for storing and retrieving persistent session information.
- Value— < DN >
You can use the special value < base > to refer to the globally configured base DN.
- Default—*o = PersistentSessions, < base >*
- Property name—`UserDataSource.repository.ldap.server.persistent.session`

Enabling Automatic Discovery of Changes in SAE Configuration Data

To use SDX Configuration Editor to enable directory eventing of SAE configuration data:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **LDAP** tab.



Enable Configuration Directory Eventing

3. Edit or accept the default value in the Enable Configuration Directory Eventing field.

See *Enable Configuration Directory Eventing Field* on page 41.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Enable Configuration Directory Eventing Field

In SDX Configuration Editor, you can edit the following fields in the LDAP pane in an SAE configuration file.

Enable Configuration Directory Eventing

- Enables automatic discovery of changes in the SAE configuration data. If this property is enabled, the SAE detects changes in its configuration and reconfigures itself.
- Value—Yes or No
- Default—Yes
- Property name—Main.auto-reconfigure

Storing Subscriber and Service Session Data

To aid in recovering from an SAE failover, the SAE stores subscriber and service session data. In releases earlier than SDX Release 6.2, the software stored subscriber and session data on the router. The SAE now stores session data in flat files on the SAE host. The SRC component that controls the storage of session data on the SAE is called the session store. The session store queues data and then writes the data to session store files on the SAE host's disk. Once the data is written to disk, it can survive a server reboot.

You can configure how the SAE stores session data for JUNOSe routers, JUNOS routing platforms, simulated routers, and *PacketCable Multimedia Specification* (PCMM) devices.

Session Store Files

Session store files are numbered flat files. When the SAE starts, the session store component creates a directory on the SAE host. Session store files are located in the directory. You can configure the size of session store files. Once the maximum size is reached, the session store creates a new file and begins writing data to the new file.

Store operations, such as adding a session to the store (put store operations) or removing a session from the store (remove store operations), are queued in a buffer before they are written to the session store file. You can configure parameters that determine when the session store writes a queue to a session store file.

Session store files are deleted if they have not been modified and if no session activity has taken place for one week. All of the data files that contain the sessions associated with a particular virtual router are deleted at the same time.

Active and Passive Session Stores

You can have a community of SAEs and duplicate session store data on each SAE in the community in case of an SAE failover. SAE communities are made up of SAEs that you configure as connected SAEs for a virtual router object. You can configure SAE communities with SDX Admin by selecting the SAE Connection tab in the VirtualRouter pane.

SAEs in a community are given the role of either active SAE or passive SAE. The active SAE keeps session data up to date within the community. Each active session store opens a Transmission Control Protocol (TCP) connection to its passive SAE. The TCP connection triggers the creation of a passive session store in that SAE. When the active session store writes operations to the session store file, it passes them to passive sessions stores on all SAEs in the community.

When you modify a community, wait for passive session stores on the new community members to be updated before you shut down the currently active SAE. Otherwise, if you add a new member to a community, and then a failover from the current active SAE to the new member is triggered immediately, the new member's session store may not have received all data from the active SAE's session store.

Session Store File Rotation

The session store periodically rotates the session store files. During rotation, the session store copies put store operations for live sessions from the oldest file to the end of the newest file. (Live sessions are sessions that have been created but not yet deleted.) It then deletes the oldest file. Sessions are rotated in batches, and you can configure the number of sessions that are rotated at the same time, and how much disk space is used by live sessions before files are rotated. No session store activity can take place while a batch of sessions is rotated.

Configuring the Session Store Feature

There are three things that you can configure for the session store feature:

- Configure session store parameters for a router or device driver. See *Configuring Session Store Parameters for a Device Driver* on page 43.
- Configure global session store parameters that are shared by all session store instances (active or passive) on the SAE. See *Configuring Global Session Store Parameters* on page 47.
- Reduce the size of session objects that the SAE sends across the network for the session store feature. See *Reducing the Size of Objects for the Session Store Feature* on page 48.

Configuring Session Store Parameters for a Device Driver

This section describes how to configure session store parameters within a router or other device driver configuration.

To use SDX Configuration Editor to configure session store parameters for a router driver or other device driver:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, and expand the driver section.

Session Store	
Maximum Queue Age [ms]	5000
Maximum Queued Operations	50
Maximum Queue Size [bytes]	51050
Maximum File Size [bytes]	25000000
Minimum Disk Space Usage	25
Rotation Batch Size	50
Maximum Session Data Bytes	10000
Disk Load Buffer Size [bytes]	1000000
Network Buffer Size [bytes]	51050
Retry Interval [ms]	300000
Communications Timeout [ms]	60000
Load Timeout [ms]	420000
Session Store Idle Timeout [ms]	3600000
Maximum Backlog Ratio	1.5
Minimum Backlog	5000000

3. Edit or accept the default values in the fields.

See *Session Store Fields* on page 44.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Session Store Fields

In SDX Configuration Editor, you can edit the following fields for a driver in the Router pane in an SAE configuration file.

Maximum Queue Age [ms]

- Maximum age that a queue of buffered store operations (such as adding a session to the store or removing a session from the store) can reach before the queue is written to a session store file.
- Value—Number of milliseconds in the range 0–2147483647. A value of –1 indicates that there is no limit. A value of zero causes the session store to write each store operation to a session store file immediately.
- Default—5000
- Property name—Router. < deviceType > .sessionStore.maxQueueAge

Maximum Queued Operations

- Number of buffered store operations that are queued before the queue is written to a session store file.
- Value—Integer in the range 0–2147483647. A value of –1 indicates that there is no limit. A value of zero causes the session store to write each store operation to a session store file immediately.
- Default—50
- Property name—Router. < deviceType > .sessionStore.maxQueueOps

Maximum Queue Size [bytes]

- Maximum size that a queue of buffered store operations can reach before the queue is written to a session store file.
- Value—Number of bytes in the range 0–2147483647
- Default—51050
- Property name—Router. < deviceType > .sessionStore.maxQueueBytes

Maximum File Size [bytes]

- Maximum size of session store files. When a file reaches this size, a new file is created.
- Value—Number of bytes in the range 0–2147483647
- Default—25000000
- Property name—Router. < deviceType > .sessionStore.maxFileBytes

Minimum Disk Space Usage

- Percentage of space in all session store files that is used by live sessions. When the percentage of space in the session store files that is used by live sessions decreases to this percentage, the oldest session store file is compacted and appended to the newest session store file, and then the oldest session store file is deleted.
- Value—Percentage of disk space in the range 1–100

- Guidelines—We recommend a range of 30–50.
- Default—40
- Property name—Router. < deviceType > .sessionStore.minDiskSpaceUsage

Rotation Batch Size

- When the oldest session store file is rotated, specifies the number of sessions that are rotated from the oldest file to the newest file at the same time. While a set of sessions is rotated, no other session store activity can take place.
- Value—Integer in the range 0–2147483647
- Default—50
- Property name—Router. < deviceType > .sessionStore.rotationBatchSize

Maximum Session Data [bytes]

- Maximum size of a single subscriber or service session. Use this parameter to reserve memory for an internal buffer.
- Value—Number of bytes in the range 0–2147483647
- Default—10000
- Property name—Router. < deviceType > .sessionStore.maxSessionDataBytes

Disk Load Buffer Size [bytes]

- Size of the buffer that is used to load all of a session store's files from disk at startup.
- Value—Number of bytes in the range 0–2147483647
- Default—1000000
- Property name—Router. < deviceType > .sessionStore.diskLoadBufferBytes

Network Buffer Size [bytes]

- Size of the buffer that holds messages or message segments that are waiting to be sent to passive session stores.
- Value—Number of bytes in the range
21 + < size of maximum session size field > – 2147483647
- Default—51050
- Property name—Router. < deviceType > .sessionStore.networkBufferBytes

Retry Interval [ms]

- Time interval between attempts by the active session store to connect to missing passive session stores.
- Value—Number of milliseconds in the range 0–2147483647
- Default—300000
- Property name—Router. < deviceType > .sessionStore.retryInterval

Communications Timeout [ms]

- Amount of time that a session store waits before closing when it is blocked from reading or writing a message. This timeout does not apply when a session store is waiting for a remote session store to load its state from disk.
- Value—Number of milliseconds in the range 0–2147483647
- Default—60000
- Property name—Router. < deviceType > .sessionStore.communicationsTimeout

Load Timeout [ms]

- Amount of time that an active session store waits for a passive session store or a passive session store waits for an active session store to load its data from disk before it closes the connection to the session store.
- Value—Number of milliseconds in the range 0–2147483647
- Default—420000
- Property name—Router. < deviceType > .sessionStore.remoteStoreLoadTimeout

Session Store Idle Timeout [ms]

- Amount of time that a passive session store waits for activity from the active session store before it closes the connection to the active session store. This timeout applies after the session store startup and initial update processes are complete.
- Value—Number of milliseconds in the range 0–2147483647
- Default—3600000
- Property name—Router. < deviceType > .sessionStore.idleTimeout

Maximum Backlog Ratio

- Along with the minimum backlog size, specifies when the active session store closes the connection to a passive session store because of a backlog of messages waiting to be sent. After the startup and initial update processes are complete, if the backlog becomes too large, the connection to the passive session store is closed. After the retry interval ends, a new connection is opened.

If the backlog of unsent operations (in bytes) divided by the total size (in bytes) of all live store operations is greater than this number, the connection is closed.
- Value—Floating point number
- Default—1.5
- Property name—Router. < deviceType > .sessionStore.backlogDeathMaxRatio

Minimum Backlog Size [bytes]

- Along with the maximum backlog ratio, specifies when the active session store closes the connection to a passive session store because of a backlog of messages waiting to be sent to the passive session store. After the startup and initial update processes are complete, if the backlog becomes too large, the connection to the passive session store is closed. After the retry interval ends, a new connection is opened.

If the maximum backlog ratio is met, the active session store does not close the connection unless the backlog of messages (in bytes) is greater than this number.

- Value—Number of bytes in the range 0–2147483647
- Default—5000000
- Property name—Router. < deviceType > .sessionStore.backlogDeathMinBehind

Configuring Global Session Store Parameters

This section describes how to configure global session store parameters that are shared by all session store instances (active or passive) on the SAE. You can also configure session store parameters within a router or other device driver configuration. See *Configuring Session Store Parameters for a Device Driver* on page 43.

To use SDX Configuration Editor to configure global session store parameters:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **Session Store** section.

The screenshot shows a configuration window titled "Session Store". It contains three rows of configuration fields:

Session Store	
Session Store IP Address	<input type="text"/> Disable
Session Store Port	<input type="text"/> Disable
Root Directory	<input type="text"/> Disable

3. Edit or accept the default values in the fields.
See *Global Session Store Fields* on page 48.
4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Global Session Store Fields

In SDX Configuration Editor, you can edit the following fields in the Session Store section of the Router pane in an SAE configuration file.

Session Store IP Address

- IP address or hostname that the session store infrastructure on this SAE uses to listen for incoming TCP connections from active session stores.
- Value—IP address. The address must be an IP address configured for the SAE host. If you do not enter an address or if you disable this field, active session stores cannot create passive session stores on this SAE.
- Guidelines—We recommend that you enter an address that is configured in a list of connected SAEs.
- Default—No value
- Property name—Router.sessionStore.ServerIp

Session Store Port

- TCP port number on which the session store infrastructure on this SAE listens for incoming connections from active session stores. Note that this field has no effect if you have not configured a session store IP address.
- Value—Port number
- Default—8820
- Property name—Router.sessionStore.ServerPort

Root Directory

- Root directory in which the session store creates files. Note that this field has no effect if you have not configured a session store IP address.
- Value—Directory name
- Default—var/sessionStore
- Property name—Router.sessionStore.rootDir

Reducing the Size of Objects for the Session Store Feature

You can use serialized data compression to reduce the size of sessions objects that the SAE sends across the network for the session store feature. Enabling this property reduces the size of objects, but increases the CPU load on the SAE.

To use SDX Configuration Editor to specify whether or not session objects are compressed:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Miscellaneous** tab.



The screenshot shows a configuration field labeled 'Compress Serialized Data' with a text input box and a dropdown arrow icon on the right.

3. Fill in the Compress Serialized Data field.
See *Compressed Serialized Data Field* on page 49.
4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Compressed Serialized Data Field

In SDX Configuration Editor, you can edit the following field in the Miscellaneous pane in an SAE configuration file.

Compress Serialized Data

- Enables or disables reducing the size of session objects (subscriber and service sessions) that the SAE sends across the network for the session store feature.
- Value—Yes or No
- Guidelines—We recommend that you do not enable this option because of the increase to the CPU load.
- Default—No
- Property name—Main.compressSerializedData

Configuring the Number of Threads for Sessions

To use SDX Configuration Editor to configure the number of threads used to handle session-related activity:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Miscellaneous** tab, and expand the **Session Job Manager** section.



3. Fill in the field.
See [Number of Threads for Sessions Field](#) on page 50.
4. Select **File > Save**.
5. Right-click the configuration file, **select SDX System Configuration > Export to LDAP Directory**.

Number of Threads for Sessions Field

In SDX Configuration Editor, you can edit the following field in the Session Job Manager section of the Miscellaneous pane in an SAE configuration file.

Number of Threads

- Sets the number of threads used for session-related activity; for example, interim accounting, subscriber and service session timeout, idle timeouts, aggregate service keepalives, and remote session monitoring.
- Value—Integer in the range 1–50
- Default—10
- Property name—SessionJobManager.numThreads

Chapter 4

Managing SAE Data with the SRC CLI

This chapter describes how to use the CLI to manage the SAE on a Solaris platform or on the C-series platform. Topics include:

- Commands to Manage SAE on page 51
- Reloading the SAE Data on page 52
- Updating Memory Usage on page 53
- Removing the Directory Blacklist on page 53
- Removing Login Registrations on page 54
- Removing Equipment Registrations on page 54
- Modifying Failover Server Parameters on page 55
- Shutting Down the Device Drivers on page 56

Commands to Manage SAE

You can use the following operational mode commands to manage SAE data:

- `clear sae directory-blacklist`
- `clear sae registered equipment`
- `clear sae registered login`
- `request sae java-garbage-collection`
- `request sae load configuration`
- `request sae load domain-map`
- `request sae load interface-classification`
- `request sae load services`
- `request sae load subscriptions`

- request sae modify device failover
- request sae shutdown device
- show sae directory-blacklist
- show sae drivers
- show sae registered equipment
- show sae registered login

For detailed information about each command, see the *SRC CLI Command Reference*.

Reloading the SAE Data

You can reload specified configuration components. You can reload the SAE server's current configuration for:

- SAE configuration
- Services
- Subscriptions
- Interface classifiers
 - Domain map

To view configuration information, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 13, Monitoring SAE Data with the SRC CLI*.

Reloading the SAE Configuration

To reload the SAE configuration data from the directory:

```
user@host> request sae load configuration
```

The new configuration takes effect immediately.

Reloading Services

To reload the services, scopes, virtual routers, policies, service mutex groups, and service schedules from the directory:

```
user@host> request sae load services
```

Related service sessions are activated, deactivated, or reactivated as needed.

Reloading Subscriptions

To reload all subscriptions from the directory:

```
user@host> request sae load subscriptions
```

Related service sessions are activated, deactivated, or reactivated as needed.

Reloading Interface Classification Scripts

To reload the interface classification scripts from the directory, and apply the result of the interface classification changes to the router:

```
user@host> request sae load interface-classification
```

Reloading Domain Maps

To reload the mapping of domain names to retailer entries:

```
user@host> request sae load domain-map
```

This mapping is made available to the SAE's subscriber classification script.

Updating Memory Usage

To ensure that changes are updated, run Java Virtual Machine (JVM) garbage collection. This process frees memory and results in more accurate Heap in Use statistics.

```
user@host> request sae java-garbage-collection
```

Removing the Directory Blacklist

To remove the directory blacklist:

1. Issue the `show sae directory-blacklist` command to view information about the directory blacklist.
2. Issue the `clear sae directory-blacklist` command to remove the directory blacklist.

Removing Login Registrations

You can delete all login registrations, or you can delete a specific registration. For information about login registrations, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 4, Configuring Subscriber-Related Properties on the SAE with the SRC CLI*.

To remove login registrations:

1. Issue the `show sae registered login` command to view the login registrations.
2. Issue the `clear sae registered login` command to remove all login registrations.

- To remove a specific registration, use the `mac-address` option and specify the media access control (MAC) address for the registration.

```
user@host> clear sae registered login mac-address mac-address
```

- To specify that no confirmation is requested before the software deletes the registration entries, use the `force` option.

```
user@host> clear sae registered login force
```

```
user@host> clear sae registered login mac-address mac-address force
```

Removing Equipment Registrations

You can delete all equipment registrations, or you can delete a specific registration. For information about equipment registrations, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 17, How Subscribers Use the Sample Residential Portal*.

To remove equipment registrations:

1. Issue the `show sae registered equipment` command to view the equipment registrations.
2. Issue the `clear sae registered equipment` command to remove all equipment registrations.

- To remove a specific registration, use the `mac-address` option and specify the media access control (MAC) address for the registration.

```
user@host> clear sae registered equipment mac-address mac-address
```

- To specify that no confirmation is requested before the software deletes the registration entries, use the `force` option.

```
user@host> clear sae registered equipment force
```

```
user@host> clear sae registered equipment mac-address mac-address force
```

Modifying Failover Server Parameters

To modify failover server parameters:

1. Issue the `show sae drivers brief` command to view the router or device instances.
2. Issue the `request sae modify device failover virtual-router-name virtual-router-name` command to modify failover server parameters.

- (Optional) To modify the IP address of an alternate SAE server to which a router can reconnect when this driver closes its connection, use the `ip-address` option. This option is not applicable to the PCMM device driver.

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name ip-address ip-address
```

- (Optional) To modify the port of an alternate SAE server to which a router can reconnect when this driver closes its connection, use the `tcp-port` option. This option is not applicable to the PCMM device driver.

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name tcp-port tcp-port
```

- (Optional) To specify whether the device driver sends its own failover IP address and port to the router when it closes its connection, use the `use-failover-server` option. This option is not applicable to the PCMM device driver.

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name use-failover-server
```

- (Optional) To specify that no confirmation is requested before the software modifies the parameters, use the `force` option.

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name force
```

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name ip-address ip-address force
```

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name tcp-port tcp-port force
```

```
user@host> request sae modify device failover virtual-router-name
virtual-router-name use-failover-server force
```

Shutting Down the Device Drivers

To shut down the specified router or device instance:

1. Issue the **show sae drivers brief** command to view the router or device instances.
2. Issue the **request sae shutdown device** command to shut down all device drivers.

- To shut down specific drivers managing a virtual router, use the **filter** option and specify all or part of the name of the virtual router.

```
user@host> request sae shutdown device filter filter
```

- To specify that no confirmation is requested before the software shuts down the device drivers, use the **force** option.

```
user@host> request sae shutdown device force
```

```
user@host> request sae shutdown device filter filter force
```

Part 2

Using Juniper Networks Routers in the SRC Network

Chapter 5

Using JUNOSe Routers in the SRC Network with the SRC CLI

This chapter describes how to use the SRC CLI to set up the SRC software and how to set up a JUNOSe router so that the router can be used in the SRC network. It also shows how to monitor the interactions between the SAE and the JUNOSe router and how to troubleshoot SRC problems on the router.

You can also use SRC configuration applications to configure the SRC software on a Solaris platform. See *Chapter 6, Using JUNOSe Routers in the SRC Network with a Solaris Platform*.

Topics in this chapter include:

- COPS Connection Between JUNOSe Routers and the SAE on page 60
- Adding JUNOSe Routers and Virtual Routers with the CLI on page 60
- Configuring the SAE to Manage JUNOSe Routers with the CLI on page 64
- Using SNMP to Retrieve Information from JUNOSe Routers on page 66
- Developing Router Initialization Scripts on page 68
- Specifying Router Initialization Scripts on the SAE with the CLI on page 70
- Accessing the Router CLI on page 72
- Starting the SRC Client on a JUNOSe Router on page 72
- Stopping the SRC Client on a JUNOSe Router on page 73
- Monitoring Interactions Between the SAE and the JUNOSe Router on page 73
- Troubleshooting Problems with Managing JUNOSe Routers on page 74

COPS Connection Between JUNOSe Routers and the SAE

Configuring the SRC client on a JUNOSe router opens a Common Open Policy Service (COPS) protocol layer connection to the SAE. When the SRC client software establishes a TCP/IP connection to the SAE, the SAE starts to manage the JUNOSe router. Subsequently, the SRC client sends configuration changes made on the JUNOSe router to the SAE, and the SAE updates SRC configurations for services and policies accordingly.

The SAE supports two versions of COPS:

- COPS usage for policy provisioning (COPS-PR)
- COPS External Data Representation Standard (COPS-XDR)

The version of COPS that you use depends on the version of COPS that your JUNOSe router supports. When you set up your JUNOSe router to work with the SAE, you enable either COPS-PR mode or COPS-XDR mode.

Adding JUNOSe Routers and Virtual Routers with the CLI

The SAE uses router and virtual router objects to manage interfaces on JUNOSe virtual routers. Each JUNOSe router in the SRC network and its virtual routers (VRs) must have a configuration.

There are two ways to add routers:

- Detect operative routers and configured JUNOSe VRs in the SRC network and add them to the configuration.
- Add each router and VR individually.

Adding Operative JUNOSe Routers and Virtual Routers

To add routers and JUNOSe VRs that are currently operative and have an operating SNMP agent:

- In operational mode, enter the following command:

```
request network discovery network network <community community>
```

where:

- *network*—Address (with or without mask) of the network to discover
- *community*—Name of the SNMP community to which the devices belong

If you add a router using the discover network feature, the software adds the IP address of the first SNMP agent on the router to respond to the discover request.

Adding Routers Individually

Use the following configuration statements to add a router:

```
shared network device name {
    description description;
    management-address management-address;
    device-type (junose| junos| pcmm| proxy);
    qos-profile [qos-profile...];
}
```

To add a router:

1. From configuration mode, access the configuration statements that configure network devices. This procedure uses `junose_boston` as the name of the router.

```
user@host# edit shared network device junose_boston
```

2. (Optional) Add a description for the router.

```
[edit shared network device junose_boston]
user@host# set description description
```

3. (Optional) Add the IP address of the router.

```
[edit shared network device junose_boston]
user@host# set management-address management-address
```

4. (Optional) Specify the type of device that you are adding.

```
[edit shared network device junose_boston]
user@host# set device-type junose
```

5. (Optional) Specify quality of service (QoS) profiles that are configured on the router.

```
[edit shared network device junose_boston]
user@host# set qos-profile [qos-profile...]
```

6. (Optional) Verify your configuration.

```
[edit shared network device junose_boston]
user@host# show
description "Juniper Networks E320";
management-address 10.10.8.27;
device-type junose;
qos-profile dhcp-default;
interface-classifier {
    rule rule-0 {
        script #;
    }
}
```

Adding Virtual Routers Individually

Use the following configuration statements to add a virtual router:

```
shared network device name virtual-router name {
    sae-connection [sae-connection...];
    snmp-read-community snmp-read-community;
    snmp-write-community snmp-write-community;
    scope [scope...];
    local-address-pools local-address-pools;
    static-address-pools static-address-pools;
    tracking-plug-in [tracking-plug-in...];
}
```

To add a virtual router:

1. From configuration mode, access the configuration statements for virtual routers. This procedure uses `junose_boston` as the name of the router and `vr1` as the name of the virtual router.

```
user@host# edit shared network device junose_boston virtual-router vr1
```

2. Specify the addresses of SAEs that can manage this router. This step is required for the SAE to work with the router.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set sae-connection [sae-connection...]
```

3. (Optional) Specify an SNMP community name for SNMP read-only operations for this VR.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set snmp-read-community snmp-read-community
```

4. (Optional) Specify an SNMP community name for SNMP write operations for this virtual router.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set snmp-write-community snmp-write-community
```

5. (Optional) Specify service scopes assigned to this virtual router. The scopes are available for subscribers connected to this virtual router for selecting customized versions of services.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set scope [scope...]
```

6. (Optional) Specify the list of IP address pools that a JUNOS virtual router currently manages and stores.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set local-address-pools local-address-pools
```

7. (Optional) Specify the list of IP address pools that a JUNOSe VR manages but does not store.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# set static-address-pools static-address-pools
```

8. (Optional) Specify the plug-ins that track interfaces that the SAE manages on this virtual router.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# tracking-plugin [tracking-plugin...]
```

9. (Optional) Verify your configuration.

```
[edit shared network device junose_boston virtual-router vr1]
user@host# show
sae-connection 192.168.10.25;
  snmp-read-community *****;
  snmp-write-community *****;
  scope POP-Boston;
local-address-pools "(10.25.8.0 10.25.20.255)";
static-address-pools "({10.30.30.0/24,10.30.30.0,10.30.30.255})";
tracking-plugin flexRadius;
```

Related Information

For additional information, see the following sources:

- For information about service scopes, see *SRC-PE Services and Policies Guide, Chapter 1, Managing Services with the SRC CLI*
- For information about local IP address pools, see *Developing Router Initialization Scripts* on page 80
- For information about tracking plug-ins, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Authorization and Accounting Plug-Ins with the CLI*

Configuring the SAE to Manage JUNOS Routers with the CLI

To set up the SAE to manage JUNOS routers, configure a router driver that specifies a COPS server that can accept COPS connections from the COPS client in JUNOS routers.

Use the following configuration statements to configure the SAE to manage JUNOS routers:

```
shared sae configuration driver junose {
  cops-server-port cops-server-port;
  backlog backlog;
  keepalive-interval keepalive-interval;
  message-timeout message-timeout;
  cops-message-maximum-length cops-message-maximum-length;
  cops-message-read-buffer-size cops-message-read-buffer-size;
  cops-message-write-buffer-size cops-message-write-buffer-size;
  pending-address-timeout pending-address-timeout;
  cops-handler-threads cops-handler-threads;
  cached-driver-expiration cached-driver-expiration;
  drop-unmanaged-interfaces-xdr-driver;
  track-unmanaged-interfaces-xdr-driver;
}
```

To configure the SAE to manage JUNOS routers:

1. From configuration mode, access the configuration statement that configures the JUNOS router driver. In this sample procedure, the JUNOS driver is configured in the west-region group.

```
user@host# edit shared sae group west-region configuration driver junose
```

2. Configure the port number of the SAE COPS server. The port number must match the configuration of the SRC client in the JUNOS router.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cops-server-port cops-server-port
```

3. Configure the number of outstanding connection attempts before connections are dropped.

```
[edit shared sae group west-region configuration driver junose]
user@host# set backlog backlog
```

4. Configure the interval between keepalive messages sent from the COPS client (the JUNOS router).

```
[edit shared sae group west-region configuration driver junose]
user@host# set keepalive-interval keepalive-interval
```

5. Configure the timeout interval in which the COPS server waits for a response to COPS requests.

```
[edit shared sae group west-region configuration driver junose]
user@host# set message-timeout message-timeout
```

6. Configure the maximum length of a COPS message.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cops-message-maximum-length cops-message-maximum-length
```

7. Configure the buffer size for receiving COPS messages from the JUNOSe client. We recommend that you use the default setting unless you are instructed to change it by Juniper Networks.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cops-message-read-buffer-size cops-message-read-buffer-size
```

8. Configure the buffer size for sending COPS messages to the JUNOSe client. We recommend that you use the default setting unless you are instructed to change it by Juniper Networks.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cops-message-write-buffer-size cops-message-read-buffer-size
```

9. Configure the maximum time that a DHCP address request remains pending.

```
[edit shared sae group west-region configuration driver junose]
user@host# set pending-address-timeout pending-address-timeout
```

10. Configure the size of the thread pool for handling unsolicited messages. These threads are shared among all JUNOSe router drivers.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cops-handler-threads cops-handler-threads
```

11. Configure the minimum amount of time to keep the state of a router driver after its COPS connection has been closed.

```
[edit shared sae group west-region configuration driver junose]
user@host# set cached-driver-expiration cached-driver-expiration
```

12. (Optional) If you are using COPS-XDR, specify whether or not the JUNOSe router driver keeps a record of unmanaged interfaces.

```
[edit shared sae group west-region configuration driver junose]
user@host# set drop-unmanaged-interfaces-xdr-driver
```

13. (Optional) Enable or disable sending of interface-tracking events for unmanaged interfaces for the XDR router driver.

```
[edit shared sae group west-region configuration driver junose]
user@host# set track-unmanaged-interfaces-xdr-driver
```

14. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration driver junose]
user@host# show
cops-server-port 3288;
backlog 50;
keepalive-interval 45;
message-timeout 120000;
cops-message-maximum-length 200000;
```

```
cops-message-read-buffer-size 30000;
cops-message-write-buffer-size 30000;
pending-address-timeout 5000;
cops-handler-threads 20;
cached-driver-expiration 600;
drop-unmanaged-interfaces-xdr-driver;
track-unmanaged-interfaces-xdr-driver;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Using SNMP to Retrieve Information from JUNOSe Routers

Some scripts in the SRC software use SNMP to get information from the router. For example, the **poolPublisher** router initialization script uses SNMP to read the IP pools.

- On the router, you can configure access to the router's SNMP server. See *Configuring the SNMP Server on the JUNOSe Router* on page 66.
- On the SAE, you can configure global default SNMP communities that are used for read and write access to the router. See *Configuring Global SNMP Communities in the SRC Software* on page 67.
- You can specify SNMP communities for each virtual router. We recommend that you specify communities for each virtual router instead of configuring global communities. See *Adding Virtual Routers Individually* on page 62.

Configuring the SNMP Server on the JUNOSe Router

Access to the SNMP server on the router by an SNMP client is governed by a proprietary SNMP community table. This table identifies communities that have read-only, read-write, or administrative permission to the SNMP Management Information Base (MIB) stored on a particular server.

When an SNMP server receives a request, the server extracts the client's IP address and the community name. The SNMP server searches the community table for a matching community.

- If a match is found, its access list name is used to validate the IP address.
 - If the access list name is null, the IP address is accepted.
 - If an invalid IP address results, an SNMP authentication error is sent to the SNMP client.
- If a match is not found, an SNMP authentication error results.

To configure the SNMP agent on the JUNOSe router:

1. Switch to the virtual router for which you want to create an SRC client.

```
host1#(config)virtual-router <vrName>
```

2. Enable the SNMP agent.

```
host1:<vrName>#(config)snmp-server
```

3. Configure at least one authorized SNMP read-write community (SNMPv1/v2c), which provides SNMP client access.

```
host1:<vrName>#(config)snmp-server community boston rw
```

4. (Optional) Configure a read-only community.

```
host1:<vrName>#(config)snmp-server public ro
```

Configuring Global SNMP Communities in the SRC Software

You can configure global default SNMP communities that are used if a VR does not exist on the router or if the community strings have not been configured for the VR.

Use the following configuration statements to configure global default SNMP communities:

```
shared sae configuration driver snmp {
    read-only-community-string read-only-community-string;
    read-write-community-string read-write-community-string;
}
```

To configure global default SNMP communities:

1. From configuration mode, access the configuration statements that configure default SNMP communities. In this sample procedure, the JUNOSe driver is configured in the west-region group.

```
user@host# edit shared sae group west-region configuration driver snmp
```

2. Configure the default SNMP community string used for read access to the router.

```
[edit shared sae group west-region configuration driver snmp]
user@host# set read-only-community-string read-only-community-string
```

3. Configure the default SNMP community string used for write access to the router.

```
[edit shared sae group west-region configuration driver snmp]
user@host# set read-write-community-string read-write-community-string
```

4. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration driver snmp]
user@host# show
read-only-community-string *****;
read-write-community-string *****;
```

Developing Router Initialization Scripts

When the SAE establishes a connection with a router, it can run a router initialization script to customize the setup of the connection. Router initialization scripts are run when the connection between a router and the SAE is established and again when the connection is dropped.

For JUNOS VRs that supply IP addresses from a local pool, a router initialization script is provided that identifies which VR supplies each IP pool and writes the information to the configuration. The SAE runs the script only when a COPS connection is established to the JUNOS router. Consequently, if you modify information about IP pools on a VR after the COPS connection is established, the SAE will not automatically register the changes, and you must update the configuration.

Table 4 describes the router initialization scripts that we provide with the SRC software in the `/opt/UMC/sae/lib` folder.

Table 4: Router Initialization Scripts

Script Name	Function	When to Use Script
iorPublisher	Publishes the IOR of the SAE into an internal part of the shared configuration so that a NIC can associate a router with an SAE.	Use with JUNOS routers that do not supply IP addresses from local pools, and with JUNOS routing platforms.
poolPublisher	Publishes the IOR of the SAE and local IP address pools in the directory so that a NIC can associate a router with an SAE and resolve the IP-to-SAE mapping.	Use with JUNOS virtual routers that supply IP addresses from local pools.

Interface Object Fields

Router initialization scripts are written in the Python programming language (www.python.org) and executed in the Jython environment (www.jython.org).

Router initialization scripts interact with the SAE through an interface object called `Ssp`. The SAE exports a number of fields through the interface object to the script and expects the script to provide the entry point to the SAE.

Table 5 describes the fields that the SAE exports.

Table 5: Exported Fields

Ssp Attribute	Description
Ssp.properties	System properties object (class: java.util.Properties)—The properties should be treated as read-only by the script.
Ssp.errorLog	Error logger—Use the SsperrorLog.println (message) to send error messages to the log.
Ssp.infoLog	Info logger—Use the Ssp.infoLog.println (message) to send informational messages to the log.
Ssp.debugLog	Debug logger—Use the Ssp.debugLog.println (message) to send debug messages to the log.

The router initialization script must set the field Ssp.routerInit to a factory function that instantiates a router initialization object:

- <VRName> —Name of the virtual router in which the COPS client has been configured, format: virtualRouterName@RouterName
- <virtuallp> —Virtual IP address of the SAE (string, dotted decimal; for example: 192.168.254.1)
- <reallp> —Real IP address of the SAE (string, dotted decimal; for example, 192.168.1.20)
- <VRIp> —IP address of the virtual router (string, dotted decimal)
- <transportVR> —Name of the virtual router used for routing the COPS connection, or None, if the COPS client is directly connected

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
               virtuallp,
               reallp,
               VRIp,
               transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a given COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a COPS server connection is established. In case of problems, an exception should be raised that leads to the termination of the COPS connection.

Required Methods

Instances of the interface object must implement the following methods:

- *setup()*—Is called when the COPS server connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the COPS connection.
- *shutdown()*—Is called when the COPS server connection to the virtual router is terminated. This method should not raise any exceptions in case of problems.

Example: Router Initialization Script

The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality. The interface class just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.println("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.println("Setup connection to VR %(vrName)s" %
                           vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.println("Shutdown connection to VR %(vrName)s" %
                           vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

Specifying Router Initialization Scripts on the SAE with the CLI

Use the following configuration statements to specify router initialization scripts for JUNOS routers:

```
shared sae configuration driver scripts {
    extension-path extension-path;
    general general;
    junose-pr junose-pr;
    junose-xdr junose-xdr;
}
```

To configure router initialization scripts for JUNOSe routers:

1. From configuration mode, access the configuration statements that configure router initialization scripts. In this sample procedure, the scripts are configured in the west-region group.

```
user@host# edit shared sae group west-region configuration driver scripts
```

2. Specify the script for JUNOSe routers when the JUNOSe driver uses COPS-PR mode when connecting to the SAE.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set junose-pr junose-pr
```

3. Specify the script for JUNOSe routers when the JUNOSe driver uses COPS-XDR mode when connecting to the SAE.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set junose-xdr junose-xdr
```

In COPS-XDR mode, the router does not send the network access server (NAS) IP address to the SAE. If your configuration requires this value, add the following line to a JUNOSe script:

import ERXnasip

When you add the **import ERXnasip** entry, the script obtains the NAS-IP address from the router through SNMP. This mechanism can affect performance, especially when the SAE manages a large number of virtual routers.

4. Configure a router initialization script that can be used for all types of routers that the SRC software supports.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set general general
```

5. Configure a path to router initialization scripts that are not in the default location, */opt/UMC/sae/lib*.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set extension-path extension-path
```

6. (Optional) Verify your router initialization script configuration.

```
[edit shared sae group west-region configuration driver scripts]
user@host# show
junose-xdr poolPublisher;
```

Accessing the Router CLI

You can access the CLIs of Juniper Networks routers through a Telnet or secure shell connection.

- To open a Telnet session to a router, use the **telnet** operational mode command. For example:

```
user@host> telnet 10.10.10.3
```

- To open a secure shell connection, use the **ssh** operational command. For example:

```
user@host> ssh host 10.10.10.3
```

Starting the SRC Client on a JUNOSe Router

JUNOSe routers use an SRC client to interact with the SAE. See *JUNOSe Broadband Access Configuration Guide* for complete information about configuring the SRC client on a JUNOSe router.

To start the SRC client:

1. Access the router CLI.
2. Access Global configuration mode.

```
host1#configure terminal
```

3. Switch to the virtual router for which you want to create an SRC client.

```
host1(config)#virtual-router <vrName>
```

4. Enable the SRC client.

To enable COPS-PR mode:

```
host1:<vrName>(config)#sscc enable cops-pr
```

To enable COPS-XDR mode:

```
host1:<vrName>(config)#sscc enable
```

5. Set the primary address from the configuration directory.

```
host1:<vrName>(config)#sscc primary address <ipAddress> port 3288
```

Stopping the SRC Client on a JUNOSe Router

JUNOSe routers use an SRC client to interact with the SAE. See *JUNOSe Broadband Access Configuration Guide* for complete information about configuring the SRC client on the JUNOSe router.

To stop the SRC client:

1. Access the router CLI.

See *Accessing the Router CLI* on page 72.

2. Access Global configuration mode.

```
host1#configure terminal
```

3. Switch to the virtual router for which you want to stop an SRC client.

```
host1(config)#virtual-router <vrName>
```

4. Disable the SRC client.

```
host1:<vrName>(config)#no ssrc enable
```

Monitoring Interactions Between the SAE and the JUNOSe Router

To monitor the connection between the router and the SAE:

- Use the **show ssrc info** command on the JUNOSe router

To display the version number of the SRC client:

- Use the **show ssrc version** command on the JUNOSe router.

See the *JUNOSe Command Reference Guide* for details about these commands.

You can also monitor the interactions between the SRC software and the router in the log files for the SAE and in the log files generated by the JUNOSe router.

- For information about configuring logging for the SAE, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 3, Configuring Logging for SRC Components with the CLI*.
- For information about configuring logging on JUNOSe routers, see *JUNOSe System Event Logging Reference Guide*.

Troubleshooting Problems with Managing JUNOSe Routers

You can troubleshoot problems with the SRC client on JUNOSe routers and with managed JUNOSe routers, interfaces, and services on the SAE.

Troubleshooting the SRC Client on JUNOSe Routers

To troubleshoot SRC problems on the router:

1. Look at the log files for the SAE and the log files generated by the SRC client on the JUNOSe router.
 - If the log files indicate a problem with specific interfaces on the router, review the configuration of the associated policies in the SRC software, and fix any errors.
 - If the log files indicate a problem with a specific service or its associated policy rules, review the configuration of the service or policies in the SRC software, and fix any errors.
 - If the log files indicate only that the SRC client is not responding, ensure that the values in the SAE configuration match the values in the SRC client configuration on the router.
2. Restart the SRC client on the JUNOSe router.

When you restart the SRC client, the SRC client removes all policies that were installed by the SRC software and reports all interfaces again.



NOTE: DHCP addresses that were managed are not reported again, so we recommend that you do not restart the SRC client if you are managing DHCP sessions.

To restart the SRC client in COPS-PR mode, enter the following commands:

```
host1:<vrName>(config)#no sssc enable
host1:<vrName>(config)#sscc enable cops-pr
```

To restart the SRC client in COPS-XDR mode, enter the following commands:

```
host1:<vrName>(config)#no sssc enable
host1:<vrName>(config)#sscc enable
```

If restarting the SRC client does not resolve the problem, rebuild the router configuration and restart the client.

Viewing the State of JUNOSe Device Drivers with the SRC CLI

To display the state of JUNOSe drivers, use the following operational mode command.

```
show sae drivers <device-name device-name> < (brief) > <maximum-results
maximum-results>
```

For example:

```
user@host> show sae drivers device-name default@dryad
JUNOSe Driver
Device name                default@dryad
Device type                junose
Device IP                  10.227.7.244
Local IP                   10.227.7.172
TransportRouter            default@dryad
Device version              7.2.0
Start time                 Tue Feb 13 14:18:44 EST 2007
Number of notifications    20
Number of processed added  14
Number of processed changed 0
Number of processed deleted 6
Number of provisioning attempt 30
Number of provisioning attempt failed 0
Number of outstanding decisions 0
Number of SAP              7
Number of PAP              1

Job Queue
Size                        0
Age (ms)                   1
Total enqueued              28
Total dequeued              28
Average job time (ms) 426

State Synchronization
Number recovered subscriber sessions 0
Number recovered service sessions 0
Number recovered interface sessions 0
Number invalid subscriber sessions 0
Number invalid service sessions 0
Number invalid interface sessions 0
Background restoration start time Tue Feb 13 14:18:49 EST 2007
Background restoration end time Tue Feb 13 14:18:49 EST 2007
Number subscriber sessions restored in background 0
Number of provisioning objects left to collect 0
Total number of provisioning objects to collect 11
Start time                 Tue Feb 13 14:18:45 EST 2007
End time                   Tue Feb 13 14:18:47 EST 2007
Number of synched contexts 7
Number of post-sync jobs 6
```

Viewing Statistics for Specific JUNOSe Device Drivers with the SRC CLI

To display statistics for a specific JUNOSe device driver, use the following operational mode command:

```
show sae statistics device <name name> < (brief) >
```

For example:

```
user@host> show sae statistics device name default@dryad
SNMP Statistics
Add notification handle time      6
Change notification handle time   0
Client ID                        default@dryad
Delete notification handle time   0
Failover IP                      0.0.0.0
Failover port                    0
Handle message time              60
Job queue age                    0
Job queue time                   4
Number message send              158
Number of added jobs             9
Number of add notifications      4
Number of change notifications    0
Number of delete notifications   0
Number of managed interfaces     4
Number of message errors         0
Number of message timeouts       0
Number of removed jobs          9
Number of user session established 0
Number of user session removed   0
Router type                      JUNOSe COPS
Up time                          172286
Using failover server            false
```

Viewing Statistics for All JUNOSe Device Drivers with the SRC CLI

To display SNMP statistics for all JUNOSe device drivers, use the following operational mode command:

```
show sae statistics device common junose-cops
```

For example:

```
user@host> show sae statistics device common junose-cops
SNMP Statistics
Driver type                      JUNOSe COPS
Number of close requests         0
Number of connections accepted   2
Number of current connections    1
Number of open requests          2
Server address                   0:0:0:0:0:0:0:0
Server port                      3288
Time since last redirect         186703
```

Viewing the State of JUNOSe Device Drivers with the C-Web Interface

If the log files indicate a problem with a specific driver, review the configuration of the associated with the JUNOSe router driver with the C-Web interface.

1. Select **SAE** from the side pane, and click **Drivers**.

The Drivers pane appears.

The screenshot shows the Juniper C-Web interface. On the left is a navigation pane with a tree structure. The 'SAE' node is selected, and the 'Drivers' sub-pane is active. The main content area contains a form with three input fields: 'Name Of Device Driver', 'Style', and 'Maximum Results'. The 'Name Of Device Driver' field has a text input box and a help text: 'Name of device drivers. Please enter: All or part of the device driver name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.' The 'Style' field has a dropdown menu with a help text: 'Output style Choices: brief: Display only virtual router names'. The 'Maximum Results' field has a text input box and a help text: 'Number of results to be displayed. Legal range: 1 .. INF Default value: 25'. Below the input fields are 'OK' and 'Reset' buttons. At the bottom of the interface, there is a copyright notice: 'Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy.' and the Juniper logo with the tagline 'Juniper Your Net.'

2. In the Name of Device Driver box, enter a full or partial device driver name for which you want to display information, or leave the box blank to display all devices. Use the format:

<virtual router name>@<router name>

3. Select an output style from the Style list.
4. In the Maximum Results box, enter the maximum number of results that you want to receive.
5. Click **OK**.

The Drivers pane displays information about the JUNOSe device driver.

Viewing Statistics for Specific JUNOS Device Drivers with the C-Web Interface

To view SNMP statistics about devices:

1. Select **SAE** from the side pane, click **Statistics**, and then click **Device**.

The Device pane appears.

Monitor Logged in as: admin About Refresh Logout

SAE > Statistics > Device

Device

Device Name

Style

OK Reset

Name of a device.
Please enter: All or part of the device name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.

Output style
Choices:
brief: Display only device names

Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy. Juniper Your Net.

2. In the Device Name box, enter a full or partial device name for which you want to display information, or leave the box blank to display all devices.
3. Select an output style from the Style list.
4. Click **OK**.

The Device pane displays statistics for all devices.

Viewing Statistics for All JUNOSe Device Drivers with the C-Web Interface

To view SNMP statistics about specific devices:

1. Select **SAE** from the side pane, click **Statistics**, click **Device**, and then click **Common**.

The Common pane appears.

Monitor

ACP

CLI

Component

Date

Disk

Interfaces...

JPS

NIC

NTP

Redirect Server

Route...

SAE

Security

System

Logged in as: admin

About Refresh Logout

SAE > Statistics > Device > Common

Common

Device Name

Type

OK Reset

Name of a device.
Please enter: All or part of the device name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.

Display SNMP statistics for a specified device driver type.
Choices:
junos: Display SNMP statistics for JUNOS router drivers
junose-cops: Display SNMP statistics for JUNOSe router drivers
packetable-cops: Display SNMP statistics for PCMM device drivers
proxy: Display SNMP statistics for third-party drivers

Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy.

Juniper Your Net.

2. In the Device Name box, enter a full or partial device name for which you want to display information, or leave the box blank to display all devices.
3. Select **junose-cops** from the Type list.
4. Click **OK**.

The Common pane displays statistics for the specified device.

Chapter 6

Using JUNOSe Routers in the SRC Network with a Solaris Platform

This chapter describes how to set up the SRC software on a Solaris platform with the SRC configuration applications that run only on Solaris platforms. It also shows how to set up a JUNOSe router so that the router can be used the SRC network. It includes information about how to monitor the interactions between the SAE and the JUNOSe router and how to troubleshoot SRC problems on the router.

You can also use the CLI that runs on Solaris platforms and the C-series platform to configure the SRC software to work with JUNOSe routers. See *Chapter 5, Using JUNOSe Routers in the SRC Network with the SRC CLI*.

Topics in this chapter include:

- COPS Connection Between JUNOSe Routers and the SAE on page 82
- Adding JUNOSe Routers and Virtual Routers on page 82
- Configuring the SAE to Manage JUNOSe Routers on page 90
- Using SNMP to Retrieve Information from JUNOSe Routers on page 93
- Developing Router Initialization Scripts on page 95
- Specifying Router Initialization Scripts on the SAE on page 98
- Updating Local IP Address Pools for JUNOSe VRs on page 99
- Accessing the Router CLI on page 103
- Starting the SRC Client on a JUNOSe Router on page 105
- Stopping the SRC Client on a JUNOSe Router on page 106
- Monitoring Interactions Between the SAE and the JUNOSe Router on page 106
- Troubleshooting the SRC Client on JUNOSe Routers on page 107

COPS Connection Between JUNOSe Routers and the SAE

Configuring the SRC client on a JUNOSe router opens a Common Open Policy Service (COPS) protocol layer connection to the SAE. When the SRC client software establishes a TCP/IP connection to the SAE, the SAE starts to manage the JUNOSe router. Subsequently, the SRC client sends configuration changes made on the JUNOSe router to the SAE, and the SAE updates SRC configurations for services and policies accordingly.

The SAE supports two versions of COPS:

- COPS usage for policy provisioning (COPS-PR)
- COPS External Data Representation Stand (COPS XDR)

The version of COPS that you use depends on the version of COPS that your JUNOSe router supports. When you set up your JUNOSe router to work with the SAE, you enable either COPS-PR mode or COPS XDR mode.

Adding JUNOSe Routers and Virtual Routers

The SAE uses router and virtual router objects in the directory to manage interfaces on JUNOSe virtual routers. Each JUNOSe router in the SRC network and its virtual routers (VRs) must appear in the directory. There are two ways to add routers to the directory:

- Use SDX Admin to detect operative routers and configured JUNOSe VRs in the SRC network and add them to the directory.
- Add each router and VR individually. You need to add routers and VRs individually if you use an LDAP client other than SDX Admin or if you want to add inoperative routers or unconfigured JUNOSe VRs.



NOTE: You must define connected SAEs for each router in the virtual router object of the directory. This step is required for the SAE to work with the router. See *Specifying the SAEs That Can Manage the Router* on page 89.

Adding Operative JUNOSe Routers and Virtual Routers

To simultaneously add to the directory routers and JUNOSe VRs that are currently operative and have an operating Simple Network Management Protocol (SNMP) agent:

1. In the SDX Admin navigation pane, select **o = Network**, and right-click.
2. Select **Discover Network**.

The Discover Network dialog box appears.

3. Enter the IP address, the prefix of the network, and the SNMP community string.
4. Click **OK**.

Objects for all routers and JUNOSe VRs that meet the criteria you specified appear under the Networks object in the navigation pane. You can modify the configuration of these objects. For information about configuring these objects, see *Adding Routers Individually* on page 83 and *Adding Virtual Routers Individually* on page 85.

Adding Routers Individually

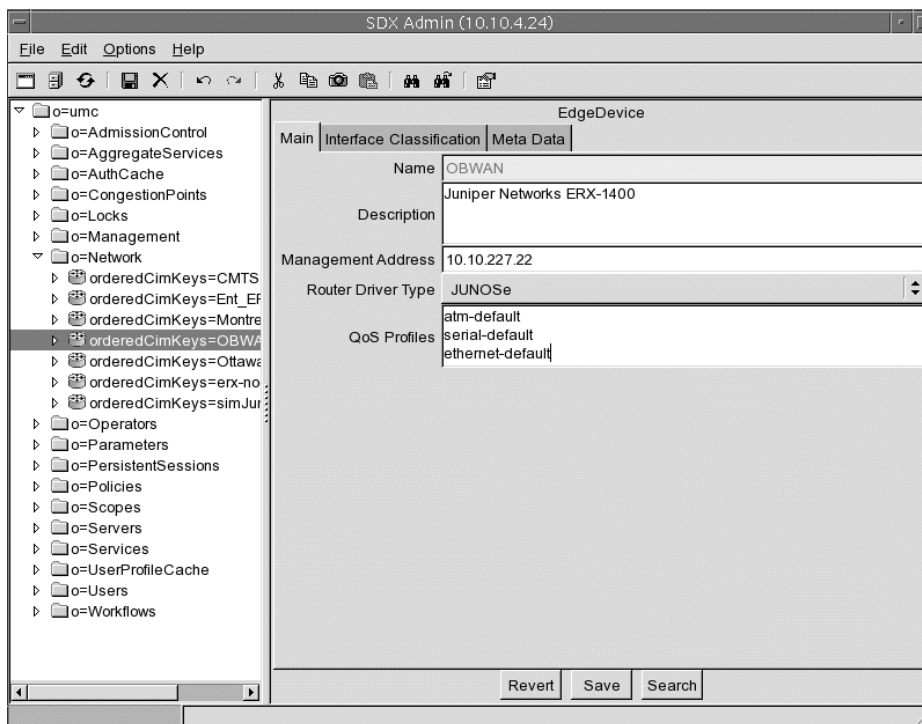
To add a single router to the directory with SDX Admin:

1. In the navigation pane, right-click the **Network** folder, and select **New > EdgeDevice**.

The New EdgeDevice dialog box appears.

2. Enter the name of the router exactly as it is configured on the router, and click **OK**.

The name of the new device appears in the navigation pane, and information about the router appears in the Main Tab of the EdgeDevice pane.



3. In the content pane, edit or accept the default values for the router fields.

See *Router Fields* on page 84.

4. Click **Save**.

Router Fields

In SDX Admin, you can modify the following fields in the content pane for a router (*orderedCimKeys* = *< EdgeDeviceName >*, *o* = *network*, *o* = *umc*).

Description

- Information about this device; keywords that the find utility uses.
- Value—Text string
- Example—ERX-1400 router located in Ottawa

Management Address

- IP address of the router. If you add a router using the discover network feature, the software automatically adds the IP address of the first SNMP agent on the router to respond to the discover request.
- Value—IP address
- Example—192.0.1.1

Router Driver Type

- Type of device that this router object will be used to manage.
- Value
 - JUNOS—JUNOS router
 - JUNOS—JUNOS routing platform
 - PCMM—CMTS device
- Default—No value

QoS Profiles

- For JUNOS routers, specifies quality of service (QoS) profiles that are configured on the router. To update this list, see *SRC-PE Solutions Guide, Chapter 1, Managing Tiered and Premium Services with QoS on JUNOS Routers*.
- Value—List of QoS profiles on separate lines
- Guideline—This field applies to JUNOS routers only
- Example—atm-default

Adding Virtual Routers Individually

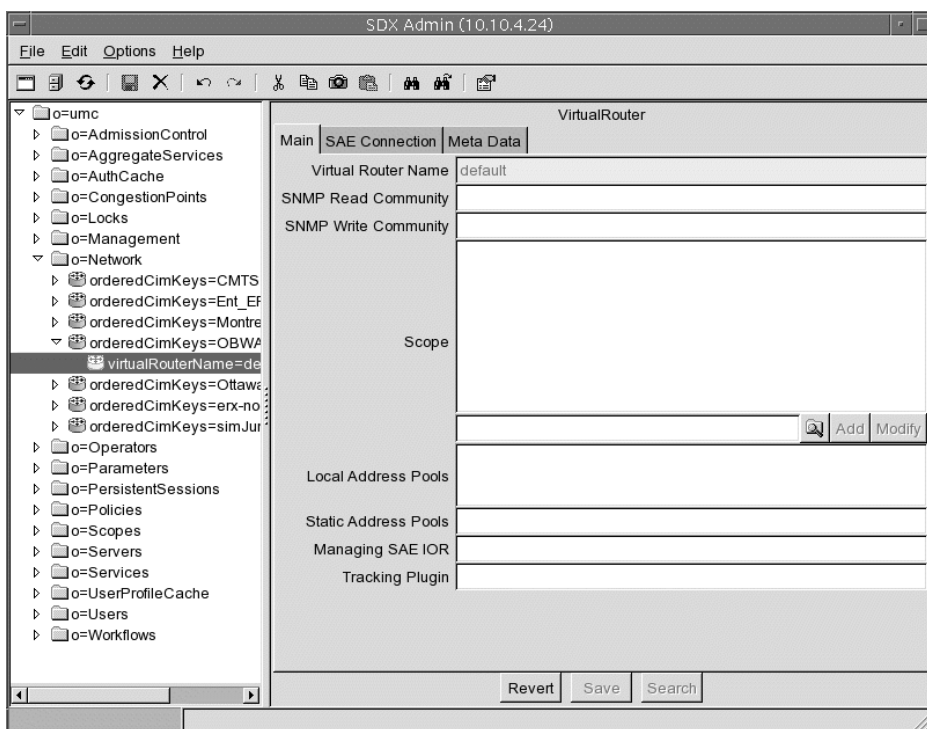
To add a VR to the directory with SDX Admin:

1. In the navigation pane, right-click the device to which you want to add the VR, and select **New > VirtualRouter**.

The New VirtualRouter dialog box appears.

2. Enter the name of the VR, and click **OK**.
 - For JUNOS routers, the name of the VR, which is case sensitive, must exactly match the name of the VR configured on the router.
 - For JUNOS routing platforms and CMTS devices, use the name default.

The name of the new VR appears in the navigation pane, and the VirtualRouter pane appears.



3. In the Main tab in the VirtualRouter pane, edit or accept the default values for the fields.

See *Virtual Router Fields* on page 86.

4. Select the **SAE Connection** tab in the VirtualRouter pane, and add SAEs that are connected to the router.

See *Specifying the SAEs That Can Manage the Router* on page 89.



NOTE: This step is required for the SAE to work with the router.

5. Click **Save**.

Virtual Router Fields

In SDX Admin, you can modify the following fields in the content pane for a virtual router (*virtualRouterName* = *<virtualRouterName = <name of virtual router>*
orderedCimKeys = *<EdgeDeviceName>* , *o = network*, *o = umc*).

SNMP Read Community

- SNMP community name associated with SNMP read-only operations for this VR.
- Value—Text string
- Example—admin

SNMP Write Community

- SNMP community name associated with SNMP write operations for this VR.
- Value—Text string
- Example—public

Scope

- Service scopes assigned to this VR.
- Value—Text string
- Example—POP-Westford

Local Address Pools

- List of IP address pools that a JUNOS VR currently manages and stores.
- Value—You can specify an unlimited number of ranges of local IP address pools for JUNOS VRs. You can specify either the first and last addresses in a range or the first IP address and a factor that indicates the start of the range. You can also specify IP addresses to exclude. Use spaces in the syntax only to separate the first and last explicit IP addresses in a range.

The IP pool syntax has the format:

```
([<ipAddressStart> <ipAddressEnd>] |  
{<ipBaseAddress>/(<mask> | <digitNumber>)(,<ipAddressExclude>)*})
```

where:

- <ipAddressStart> —First IP address (version 4 or 6) in a range
- <ipAddressEnd> —Last IP address (version 4 or 6) in a range
- <ipBaseAddress> —Network base address
- <mask> —IP address mask
- <digitNumber> —Integer specifying the number of significant digits of the first IP address in the range
- <ipAddressExclude> —List of IP addresses to be excluded from the range
- |—Choice of expression; choose either the expression to the left or the expression to the right of this symbol
- *—Zero or more instances of the preceding group
- Guidelines—If you do not configure the **PoolPublisher** router initialization scripts for a JUNOSe router, configure this field for the JUNOSe VR.
- Default—No value
- Example—This example shows four ranges for the IP address pool.
 ([10.10.10.5 10.10.10.250]
 {10.20.20.0/24}
 {10.21.0.0/255.255.0.0}
 {10.20.30.0/24,10.20.30.1})
- The first range (a simple range) specifies all the IP addresses between the two IP addresses 10.10.10.5 and 10.10.10.250.
- The second range specifies all the IP addresses in the range 10.20.20.0 to 10.20.20.255.
- The third range uses a network mask to specify all the IP addresses in the range 10.21.0.0 to 10.21.255.255.
- The fourth range specifies all the addresses of the network 10.20.30.0 to 10.20.30.255, excluding the address 10.20.30.1.

Static Address Pools

- List of IP address pools that a JUNOSe VR manages but does not store. You can configure these address pools only in the SRC software.
- Value—See the field Local Address Pools.
- Guidelines—Configure this field on JUNOSe and CMTS VRs only.
- Default—No value
- Example—([10.10.10.5 10.10.10.250] {10.20.20.0/24})

Managing SAE IOR

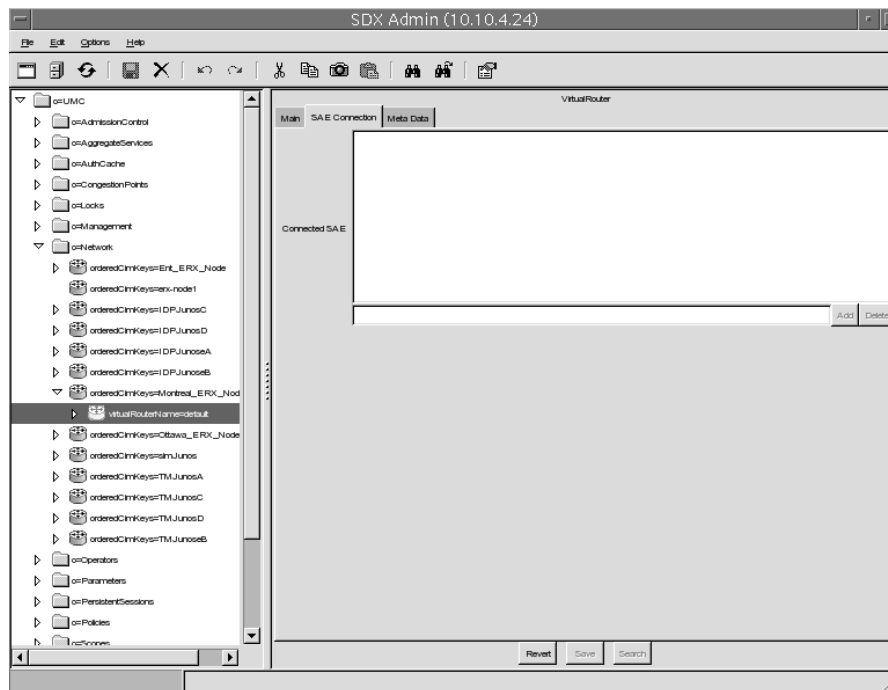
- Common Object Request Broker Architecture (CORBA) reference for the SAE managing this VR.
- Value—One of the following items:
 - The actual CORBA reference for the SAE
 - The absolute path to the interoperable object reference (IOR) file
 - A corbaloc URL in the form corbaloc:: <host > :8801/SAE
 - <host > is the name or IP address of the SAE host.
- Default—No value
- Guidelines—The **PoolPublisher** and **IorPublisher** router initialization scripts provide this information when the router connects to the SAE. If you do not use one of these router initialization scripts, enter a value in this field.
- Example—One of the following items:
 - Absolute path—`/opt/UMC/sae/var/run/sae.ior`
 - corbaloc URL—`corbaloc::boston:8801/SAE`
 - Actual IOR—`IOR:0000000000000002438444C3A736D67742E6A756E697...`

Tracking Plug-in

- Plug-ins that track interfaces that the SAE manages on this VR. The SAE calls these plug-in instances for every interface it manages. The SAE calls these plug-ins after an interface comes up, when new policies are installed on the interface, and when the interface goes down.
- Value—Comma-separated list of plug-in instances
- Guidelines—Enter plug-in instances and network information collector (NIC) SAE plug-in agents that are specific to this VR.
- Default—No value
- Example—`nicsae, flexRadius`

Specifying the SAEs That Can Manage the Router

You must add the addresses of SAEs that can manage this router. This step is required for the SAE to work with the router. To add the SAEs, select the SAE Connection tab in the VirtualRouter pane.



Adding an SAE

To add an SAE:

1. Type the IP address of the SAE in the field below the Connected SAE box.
2. Click **Add**.

Modifying an SAE Address

To modify an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Modify the IP address in the field below the Connected SAE box.
3. Click **Modify**.

Deleting an SAE Address

To delete an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Remove the IP address from the field below the Connected SAE box.
3. Click **Delete**.

Connected SAE



- SAEs that are connected to the router or CMTS device.
- Value—IP addresses
- Default—No value

Configuring the SAE to Manage JUNOSe Routers

To set up the SAE to manage JUNOSe routers, you need to configure a router driver that specifies the COPS connection between the SAE COPS server and the COPS client in the JUNOSe router.

To use SDX Configuration Editor to configure a JUNOSe router driver:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **JUNOSe Router Driver** section.

JUNOSe Router Driver	
COPS Server Port	3288
Backlog	50
Keepalive Interval [s]	45
Message Timeout [ms]	120000
COPS Message Maximum Length [bytes]	200000
COPS Message Read Buffer Size [bytes]	30000
COPS Message Write Buffer Size [bytes]	30000
Pending Address Timeout [ms]	5000
Number of COPS Handler Threads	20
Cached driver expiration	600
Drop Unmanaged Interfaces for the JUNOSe XDR Driver	No 
Track Unmanaged Interfaces for XDR Driver	No 

3. Edit or accept the default values in the fields.

See *JUNOSe Router Driver Fields* on page 91.

4. You can also configure a session store for the JUNOSe router driver.

See *Storing Subscriber and Service Session Data* on page 41.

5. Select **File > Save**.

6. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

JUNOSe Router Driver Fields

In SDX Configuration Editor, you can edit the following fields in the JUNOSe Router Driver section of the Router pane in an SAE configuration file.

COPS Server Port

- Port number of the SAE COPS server.
- Value—Port number that matches the configuration of the SRC client in the JUNOSe router
- Default—3288
- Property name—Router:junose.server_port

Backlog

- Number of connection attempts before connections are dropped.
- Value—Integer
- Default—50
- Property name—Router:junose.backlog_connections

Keepalive Interval [s]

- Interval between keepalive messages sent from the COPS client (the JUNOSe router). The COPS client monitors the COPS connection by sending keepalive messages at random intervals between one-fourth and three-fourths of the specified interval. If the client does not receive the expected keepalive answer within the specified timeout, the client terminates the connection.
- Value—Number of seconds in the range 0–32768. A value of 0 means that timeout is disabled.
- Guidelines—A short interval results in a high load on the COPS interface. A long interval results in a long time before a COPS failure is detected.
- Default—45
- Property name—Router:junose.keepalive

Message Timeout [ms]

- Timeout interval in which the COPS server waits for a response to COPS requests. Under a high load the router may not be able to respond fast enough to COPS requests. Change this value only if a high number of COPS timeout events appear in the error log.
- Value—Number of milliseconds
- Default—60000
- Property name—Router:junos.message_timeout

COPS Message Maximum Length [bytes]

- Maximum length of a COPS message.
- Value—Number of bytes in the range 4 bytes to 2 GB
- Guidelines—We recommend that you use the default setting.
- Default—200000
- Property name—Router:junos.message_max_length

COPS Message Read Buffer Size [bytes]

- Buffer size for receiving COPS messages from the JUNOS client.
- Value—Number of bytes in the range 4 bytes to 2 GB
- Guidelines—We recommend that you use the default setting unless you are instructed to change it by Juniper Networks engineers.
- Default—30000
- Property name—Router:junos.message_read_buffer_size

COPS Message Write Buffer Size [bytes]

- Buffer size for sending COPS messages to the JUNOS client.
- Value—Number of bytes in the range 4 bytes to 2 GB
- Guidelines—We recommend that you use the default setting unless you are instructed to change it by Juniper Networks engineers
- Default—30000
- Property name—Router:junos.message_write_buffer_size

Pending Address Timeout [ms]

- Maximum time that an address request remains pending.
- Value—Number of milliseconds
- Guidelines—Realistic values are in the range 1000–15000 (5 seconds to 15 seconds).
- Default—5000
- Property name—Router:junos.pending_address_timeout

Number of COPS Handler Threads

- Size of the thread pool for handling unsolicited messages. These threads are shared among all JUNOSe router drivers.
- Value—Number of threads
- Default—20
- Property name—Router.junose.handler_threads

Cache driver expiration

- Minimum amount of time to keep the state of a router driver after its COPS connection has been closed.
- Value—Number of seconds in the range 0–2147483647
- Default—600
- Property name—Router.junose.cachedDriverExpiration

Drop Unmanaged Interfaces for the JUNOSe XDR Driver

- Specifies whether or not the JUNOSe router driver keeps a record of unmanaged interfaces.
- Value
 - Yes—The router driver does not keep a record of unmanaged interfaces. With this setting, next interface rules may not work properly.
 - No—The router driver keeps a record of unmanaged interfaces.
- Default—No
- Property name—Router.junose.drop_unmanaged_xdr

Using SNMP to Retrieve Information from JUNOSe Routers

Some scripts in the SRC software use SNMP to get information from the router. For example, the **poolPublisher** router initialization script uses SNMP to read the IP pools.

- On the router, you can configure access to the router's SNMP server. See *Configuring the SNMP Server on the JUNOSe Router* on page 93.
- On the SAE, you can configure global default SNMP communities that are used for read and write access to the router. See *Configuring Global SNMP Communities in the SRC Software* on page 94.
- In the directory, you can specify SNMP communities for each virtual router. We recommend that you specify communities for each virtual router instead of global communities. See *Adding Virtual Routers Individually* on page 85.

Configuring the SNMP Server on the JUNOSe Router

Access to the SNMP server on the router by an SNMP client is governed by a proprietary SNMP community table. This table identifies communities that have read-only, read-write, or administrative permission to the SNMP Management Information Base (MIB) stored on a particular server.

When an SNMP server receives a request, the server extracts the client's IP address and the community name. The SNMP server searches the community table for a matching community.

- If a match is found, its access list name is used to validate the IP address.
 - If the access list name is null, the IP address is accepted.
 - If an invalid IP address results, an SNMP authentication error is sent to the SNMP client.
- If a match is not found, an SNMP authentication error results.

To configure the SNMP agent on the JUNOSe router:

1. Switch to the virtual router for which you want to create an SRC client.

```
host1#(config)virtual-router <vrName>
```

2. Enable the SNMP agent.

```
host1:<vrName>#(config)snmp-server
```

3. Configure at least one authorized SNMP read-write community (SNMPv1/v2c), which provides SNMP client access.

```
host1:<vrName>(config)#snmp-server community boston rw
```


4. (Optional) Configure a read-only community.

```
host1:<vrName>#(config)snmp-server public ro
```

Configuring Global SNMP Communities in the SRC Software

You can configure global default SNMP communities that are used if a VR does not exist on the router or the community strings have not been configured for the VR. To use SDX Configuration Editor to configure global default SNMP communities:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **SNMP** section.



The screenshot shows a configuration window for the 'SNMP' section. It contains two input fields: 'Read-Only Community String' and 'Read-Write Community String'. Both fields have a default value of '*****' and a 'Show' button to the right of each field.

3. Edit or accept the default values in the fields.

See Global SNMP Community Fields on page 95.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Global SNMP Community Fields

In SDX Configuration Editor, you can edit the following fields in the Router pane in an SAE configuration file.

Read-Only Community String

- Default SNMP community string used for read access to the router.
- Value—SNMP community string that matches a read-only community string configured on the router
- Default—Public
- Property name—Router.read-only.community.string

Read-Write Community String

- Default SNMP community string used for write access to the router.
- Value—SNMP community string that matches a read-write community string configured on the router
- Default—Private
- Property name—Router.read-write.community.string

Developing Router Initialization Scripts

When the SAE establishes a connection with a router, it can run a router initialization script to customize the setup of the connection. Router initialization scripts are run when the connection between a router and the SAE is established and again when the connection is dropped.

For JUNOSe VRs that supply IP addresses from a local pool, a router initialization script is provided that identifies which VR supplies each IP pool and writes the information to the directory. The SAE runs the script only when a COPS connection is established to the JUNOSe router. Consequently, if you modify information about IP pools on a VR after the COPS connection is established, the SAE will not automatically register the changes, and you must update the directory.

Table 6 describes the router initialization scripts that we provide with the SRC software in the `/opt/UMC/sae/lib` directory.

Table 6: Router Initialization Scripts

Script Name	Function	When to Use Script
IorPublisher	Publishes the IOR of the SAE in the directory so that a NIC can associate a router with an SAE.	Use with JUNOSe routers that do not supply IP addresses from local pools, and JUNOS routing platforms.
poolPublisher	Publishes the IOR of the SAE and local IP address pools in the directory so that a NIC can associate a router with an SAE and resolve the IP-to-SAE mapping.	Use with JUNOSe virtual routers that supply IP addresses from local pools.

Interface Object Fields

Router initialization scripts interact with the SAE through an interface object called `Ssp`. The SAE exports a number of fields through the interface object to the script and expects the script to provide the entry point to the SAE.

Table 7 describes the fields that the SAE exports.

Table 7: Exported Fields

Ssp Attribute	Description
<code>Ssp.properties</code>	System properties object (class: <code>java.util.Properties</code>)—The properties should be treated as read-only by the script.
<code>Ssp.errorLog</code>	Error logger—Use the <code>Ssp.errorLog.println (message)</code> to send error messages to the log.
<code>Ssp.infoLog</code>	Info logger—Use the <code>Ssp.infoLog.println (message)</code> to send informational messages to the log.
<code>Ssp.debugLog</code>	Debug logger—Use the <code>Ssp.debugLog.println (message)</code> to send debug messages to the log.

The router initialization script must set the field `Ssp.routerInit` to a factory function that instantiates a router initialization object:

- `<VRName>` —Name of the virtual router in which the COPS client has been configured, format: `virtualRouterName@RouterName`
- `<virtualIp>` —Virtual IP address of the SAE (string, dotted decimal; for example: `192.168.254.1`)
- `<realIp>` —Real IP address of the SAE (string, dotted decimal; for example, `192.168.1.20`)
- `<VRIp>` —IP address of the virtual router (string, dotted decimal)
- `<transportVR>` —Name of the virtual router used for routing the COPS connection, or `None`, if the COPS client is directly connected

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
virtualIp,
realIp,
VRIp,
transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a given COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a COPS server connection is established. In case of problems, an exception should be raised that leads to the termination of the COPS connection.

Required Methods

Instances of the interface object must implement the following methods:

- *setup()*—Is called when the COPS server connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the COPS connection.
- *shutdown()*—Is called when the COPS server connection is terminated to the virtual router. This method should not raise any exceptions in case of problems.

Example: Router Initialization Script

The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality. The interface class just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.println("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.println("Setup connection to VR %(vrName)s" %
                             vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.println("Shutdown connection to VR %(vrName)s" %
                             vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

Specifying Router Initialization Scripts on the SAE

To use SDX Configuration Editor to specify router initialization scripts:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **Router Scripts** section.

Router Scripts	
Extension Path	<input type="text"/>
General Script	<input type="text"/>
JUNOS Script	<input type="text"/>
JUNOSe Script	<input type="text"/>
JUNOSe Script (XDR)	<input type="text"/>

3. Edit or accept the default values in the fields.
See *JUNOSe Router Script Fields* on page 98.
4. Select File > Save.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

JUNOSe Router Script Fields

In SDX Configuration Editor, you can edit the following fields in the Router pane in an SAE configuration file.

Extension Path

- Path to router initialization scripts that are not in the default location, */opt/UMC/sae/lib*.
- Value—List of paths separated by semicolons (;)
- Default—No value
- Property name—Extension.path

General Script

- Router initialization script that can be used for all types of routers that the SRC software supports. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Default—No value
- Property name—Router.script.*

JUNOSe Script

- Router initialization script for JUNOSe routers when the JUNOSe driver uses COPS-PR mode when connecting to the SAE. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Default—No value
- Property name—Router.script.junos

JUNOSe Script (XDR)

- Router initialization script for JUNOSe routers when the JUNOSe driver uses XDR mode when connecting to the SAE. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Guidelines—In COPS XDR mode, the router does not send the network access server (NAS) IP address to the SAE. If your configuration requires this value, add the following line to a JUNOSe script:

import ERXnasip

When you add the **import ERXnasip** entry, the script obtains the NAS-IP address from the router through SNMP. This mechanism can affect performance, especially when the SAE manages a large number of virtual routers.

- Default—Unspecified
- Examples—iorPublisher, poolPublisher
- Property name—Router.script.junos_xdr

Updating Local IP Address Pools for JUNOSe VRs

When you reconfigure local IP address pools on a JUNOSe VR, you must update in the directory the local IP addresses that the VR provides.

Before you update local IP address pools, make sure that:

- The JUNOSe router and VR appear in the directory.
- The VR has an operating SNMP agent.
- The host that supports SDX Admin or the SAE can communicate with the VR through SNMP.
- You have write permissions for the *o = Network* subtree.

There are two ways to add routers to the directory:

- SDX Admin—Updates on VR at a time.
- The **poolRepublish** command—simultaneously updates any number of VRs in the same directory.

Updating Local IP Address Pools with SDX Admin

To allow updates of IP address pools with SDX Admin, the host that supports SDX Admin must be able to communicate with the VR through SNMP. To update local IP address pools for a VR in the directory with SDX Admin:

1. In the navigation pane, expand **o = Network**.
2. In the navigation pane, expand the object for the router on which the VR is configured.
3. Right-click the object for the VR in the navigation pane.
4. Select **Update IP Pools**.

The SDX Admin dialog box appears.

5. Enter the IP address for the VR, enter the SNMP community if the default value is incorrect, and click **OK**.

SDX Admin updates the local IP addresses for the VR in the directory and displays the information in the Local IP Address field of the Main tab in the VirtualRouter pane.

Updating Local IP Address Pools with the poolRepublish Command

You can use the **poolRepublish** command on the SAE host to update local IP address pools. You can specify multiple VRs with the **poolRepublish** command that use the same SNMP read community. For each VR you must specify the name of the VR, the name of the JUNOS router on which it is configured, the VR's corresponding IP address, and the directory connection.

To update local IP addresses using the **poolRepublish** command:

1. On the SAE host, access the folder */opt/UMC/sae/etc*.

```
cd /opt/UMC/sae/etc
```

2. Run the command.

```
./poolRepublish -v vr1@erx1 -i 192.0.2.1 -v vr2@erx2 -i 192.0.2.3 -h 192.0.2.5  
-w admin123 -D cn=umcAdmin,o=umc -b o=Network,o=umc -c public
```

The software updates and displays the local IP address pools for each VR you specified.

```
vr1@erx1 pools: ([10.227.11.242 10.227.11.250][10.227.11.226  
10.227.11.239]{10.227.11.208/255.255.255.240}{10.227.11.240/255.255.2  
55.240}{10.227.11.224/255.255.255.240})  
vr2@erx2: ([10.227.12.242 10.227.12.250][10.227.12.226  
10.227.12.239]{10.227.12.208/255.255.255.240}{10.227.12.240/255.255.2  
55.240}{10.227.12.224/255.255.255.240})
```

Syntax of poolRepublish Command

The syntax for the poolRepublish command is:

```
poolRepublish { { -v <vrName> @ <routerName> -i <ipAddress> } *  
-h <host> -b <baseDn> -D <bindDN> -w <password>  
-c <readCommunity> ] | -H }
```

<vrName>

- Name of the VR.
- Value—Text string (value is case sensitive and must match the name in the JUNOSe configuration)
- Guideline—You must enter a value for this property.
- Example—vr-boston

<routerName>

- Name of JUNOSe router on which VR is configured.
- Value—Text string (value is case sensitive and must match the name in the JUNOSe configuration)
- Example—erx1

<ipAddress>

- VR's IP address.
- Value—IP address or text string
- Example—192.0.2.1

<host>

- IP address or name of the host that supports the directory.
- Value—IP address or text string
- Example—192.0.2.2 or ottawa

<baseDn>

- DN of the root of the tree in the directory.
- Value—DN
- Example—*o = Network, o = umc*

<bindDn>

- DN of the username for authentication with the directory server.
- Value—DN
- Example—*cn = umcAdmin, o = umc*

<password>

- Password for authentication with the directory server.
- Value—Text string
- Example—Admin123

<readCommunity>

- Name of the SNMP read community for the VR. If the SNMP read community for a VR is defined in the directory, you do not need to specify this value.
- Value—Text string
- Example—public

-H

- Displays help for this tool.

Troubleshooting the poolRepublish Command

You must specify the correct arguments for the **poolRepublish** command. In addition, the specified router and directory must be available for the command to run successfully.

If no SNMP read community is configured in the directory for the VR and you do not specify this value when you run the **poolRepublish** command, you will see the following error message:

```
Could not perform ip pools update due to No 'snmpReadCommunity' attribute is
provided for virtual router: vr1@bigfoot
```

If you run the **poolRepublish** command again and supply this SNMP read community, the command should run correctly.

Accessing the Router CLI

You can access the CLIs of Juniper Networks routers from Policy Editor and from SDX Admin through a Telnet or SSH connection. This access allows you to display and change the configuration of the router.

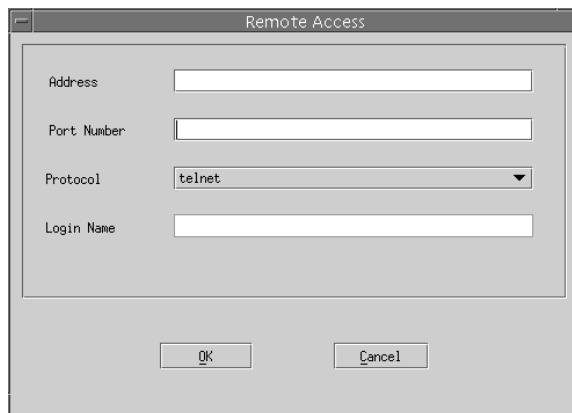
You must have the Telnet or SSH applications installed and available to Policy Editor or SDX Admin. You can open multiple Telnet or SSH sessions.

Using Policy Editor

To access a router from Policy Editor:

1. In the Policy Editor window, click **Tools** in the menu bar; then click **Manage**.

The Remote Access dialog box appears.


 A screenshot of the 'Remote Access' dialog box. It has a title bar with the text 'Remote Access'. Inside the dialog, there are four labeled text input fields: 'Address', 'Port Number', 'Protocol', and 'Login Name'. The 'Protocol' field is a dropdown menu with 'telnet' selected. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel'.

2. Fill in the Remote Access fields, and click **OK**.

See *Remote Access Fields* on page 104.

A Telnet or an SSH window with a CLI prompt appears.

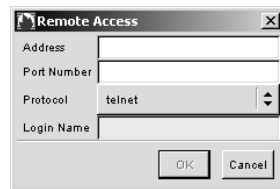
Using SDX Admin

To access a router from SDX Admin:

1. In the navigation pane, expand **o = Network**.
2. Select the router to which you want to connect, and right-click.

3. Select **Manage**.

The Remote Access dialog box appears.



4. Fill in the Remote Access fields, and click **OK**.

See *Remote Access Fields* on page 104.

A Telnet or an SSH window with a CLI prompt appears.

Remote Access Fields

In Policy Editor, you can edit the following fields in the Remote Access dialog box, in the Tools > Manage menu.

In SDX Admin, you can edit the following fields in the Remote Access dialog box by right-clicking on the router object, and selecting Manage.

Address

- IP address or hostname of the router.
- Value—IP address
- Default—No value
- Example—192.0.2.1

Port Number

- TCP port over which you want to connect to the router.
- Value—TCP port
- Default—No value
- Example—22

Protocol

- Type of connection
- Value—telnet | ssh
- Default—telnet
- Example—ssh

Login Name

- Login name for SSH connections.
- Value—Text string
- Default—No value
- Guideline—You must enter a value for this property.
- Example—admin

Starting the SRC Client on a JUNOSe Router

JUNOSe routers use an SRC client to interact with the SAE. See *JUNOSe Broadband Access Configuration Guide* for complete information about configuring the SRC client on the JUNOSe router.

To start the SRC client:

1. Access the router CLI.
2. Access Global configuration mode.

```
host1#configure terminal
```

3. Switch to the virtual router for which you want to create an SRC client.

```
host1(config)#virtual-router <vrName>
```

4. Enable the SRC client.

To enable COPS-PR mode:

```
host1:<vrName>(config)#sscc enable cops-pr
```

To enable COPS-XDR mode:

```
host1:<vrName>(config)#sscc enable
```

5. Set the primary address from the configuration directory.

```
host1:<vrName>(config)#sscc primary address <ipAddress> port 3288
```

Stopping the SRC Client on a JUNOS Router

JUNOS routers use an SRC client to interact with the SAE. See *JUNOS Broadband Access Configuration Guide* for complete information about configuring the SRC client on the JUNOS router.

To stop the SRC client:

1. Access the router CLI.

See *Accessing the Router CLI* on page 103.

2. Access Global configuration mode.

```
host1#configure terminal
```

3. Switch to the virtual router for which you want to stop an SRC client.

```
host1(config)#virtual-router <vrName>
```

4. Disable the SRC client.

```
host1:<vrName>(config)#no ssrc enable
```

Monitoring Interactions Between the SAE and the JUNOS Router

To monitor the connection between the router and the SAE:

- Use the **show ssrc info** command on the JUNOS router

To display the version number of the SRC client:

- Use the **show ssrc version** command on the JUNOS router.

See the *JUNOS Command Reference Guides* for details about these commands.

You can also monitor the interactions between the SRC software and the router in the log files for the SAE and in the log files generated by the JUNOS router. For information about configuring logging for the SAE, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 4, Configuring Logging for SRC Components on a Solaris Platform*. For information about configuring logging on JUNOS routers, see the *JUNOS System Event Logging Reference Guide*.

Troubleshooting the SRC Client on JUNOSe Routers

To troubleshoot SRC problems on the router:

1. Look at the log files for the SAE and the log files generated by the SRC client on the JUNOSe router.
 - If the log files indicate a problem with specific interfaces on the router, review the configuration of the associated policies in the SRC software, and fix any errors.
 - If the log files indicate a problem with a specific service or its associated policy rules, review the configuration of the service or policies in the SRC software, and fix any errors.
 - If the log files indicate only that the SRC client is not responding, ensure that the values in the SAE configuration match the values in the SRC client configuration on the router.
2. Restart the SRC client on the JUNOSe router.

When you restart the SRC client, the SRC client removes all policies that were installed by the SRC software and reports all interfaces again.



NOTE: DHCP addresses that were managed are not reported again, so we recommend that you do not restart the SRC client if you are managing DHCP sessions.

To restart the SRC client in COPS-PR mode, enter the following commands:

```
host1:<vrName>(config)#no ssrc enable
host1:<vrName>(config)#sscc enable cops-pr
```

To restart the SRC client in COPS XDR mode, enter the following commands:

```
host1:<vrName>(config)#no ssrc enable
host1:<vrName>(config)#sscc enable
```

If restarting the SRC client does not resolve the problem, rebuild the router configuration and restart the client.

Chapter 7

Using JUNOS Routing Platforms in the SRC Network with the SRC CLI

This chapter describes how to use the SRC CLI to set up the SRC software and how to set up JUNOS routing platforms so that the routing platforms can be used the SRC network. It also shows how to monitor the interactions between the SAE and JUNOS routing platforms and how to troubleshoot SRC problems on JUNOS routing platforms.

You can also use SRC configuration applications to configure the SRC software on a Solaris platform. See *Chapter 8, Using JUNOS Routing Platforms in the SRC Network with a Solaris Platform*.

Topics in this chapter include:

- BEEP Connection Between JUNOS Routing Platforms and the SAE on page 110
- Adding JUNOS Routing Platforms and Virtual Routers on page 110
- Configuring the SAE to Manage JUNOS Routing Platforms on page 113
- Configuring Secure Connections Between the SAE and JUNOS Routing Platforms on page 115
- Checking Changes to the JUNOS Configuration on page 121
- Using SNMP to Retrieve Information from JUNOS Routing Platforms on page 122
- Developing Router Initialization Scripts on page 123
- Specifying Router Initialization Scripts on the SAE on page 125
- Accessing the Router CLI on page 126
- Configuring JUNOS Routing Platforms to Interact with the SAE on page 126
- Disabling Interactions Between the SAE and JUNOS Routing Platforms on page 128

- Monitoring Interactions Between the SAE and JUNOS Routing Platforms on page 128
- Troubleshooting Problems with the SRC Software Process on page 129

BEEP Connection Between JUNOS Routing Platforms and the SAE

For information about which JUNOS routing platforms and releases a particular SRC release supports, see the SRC *Release Notes*.

The SAE interacts with a JUNOS software process, referred to as the SRC software process in this documentation, on the JUNOS routing platform. The SAE and the SRC software process communicate using the Blocks Extensible Exchange Protocol (BEEP). You can secure the BEEP connection by using Transport Layer Security (TLS).

When the SRC software process establishes a BEEP session for the SAE, the SAE configures an interface on the JUNOS routing platform. The SAE builds the configuration for an interface using the policies stored in the directory. If the policies are subsequently modified, the SAE builds a new configuration and reconfigures the interface on the JUNOS routing platform. The JUNOS routing platform stores data about interfaces and services that the SAE manages in a configuration group called *sdx*. You must create this configuration group on the JUNOS routing platform.

Adding JUNOS Routing Platforms and Virtual Routers

On JUNOS routing platforms, the SAE manages interfaces. The SRC software associates a virtual router called *default* with each JUNOS routing platform. Each JUNOS routing platform in the SRC network and its associated virtual router (VR) called *default* must appear in the directory. The VRs are not actually configured on the JUNOS routing platform; the VR in the directory provides a way for the SAE to manage the interfaces on the JUNOS routing platform.

There are two ways to add routers:

- Detect operative routers and configured JUNOS VRs in the SRC network and add them to the configuration.
- Add each router and VR individually.

Adding Operative JUNOS Routing Platforms

To add to the directory routers and JUNOS VRs that are currently operative and have an operating SNMP agent:

- In operational mode, enter the following command:

```
request network discovery network network <community community>
```

where:

- *network*—Address (with or without mask) of the network to discover
- *community*—Name of the SNMP community to which the devices belong

If you add a router using the discover network feature, the software adds the IP address of the first SNMP agent on the router to respond to the discover request.

Adding Routers Individually

Use the following configuration statements to add a router device:

```
shared network device name {
  description description;
  management-address management-address;
  device-type (junose| junos| pcmm| proxy);
  qos-profile [qos-profile...];
}
```

To add a router device:

1. From configuration mode, access the configuration statements that configure network devices. This procedure uses `junos_boston` as the name of the router.

```
user@host# edit shared network device junos_boston
```

2. (Optional) Add a description for the router.

```
[edit shared network device junos_boston]
user@host# set description description
```

3. (Optional) Add the IP address of the router.

```
[edit shared network device junos_boston]
user@host# set management-address management-address
```

4. (Optional) Specify the type of device that you are adding.

```
[edit shared network device junos_boston]
user@host# set device-type junos
```

5. (Optional) Verify your configuration.

```
[edit shared network device junos_boston]
user@host# show
description "This is a core-facing JUNOS router.";
management-address 10.117.8.32;
device-type junos;
interface-classifier {
  rule rule-0 {
    script #;
  }
}
```

Adding Virtual Routers Individually

Use the following configuration statements to add a virtual router:

```
shared network device name virtual-router name {
  sae-connection [sae-connection...];
  snmp-read-community snmp-read-community;
  snmp-write-community snmp-write-community;
  scope [scope...];
  tracking-plugin [tracking-plugin...];
}
```

To add a virtual router:

1. From configuration mode, access the configuration statements for virtual routers. This procedure uses `junos_Boston` as the name of the router. For JUNOS routing platforms, use the name `default` for the virtual router.

```
user@host# edit shared network device junos_boston virtual-router default
```

2. Specify the addresses of SAEs that can manage this router. This step is required for the SAE to work with the router.

```
[edit shared network device junos_boston virtual-router default]
user@host# set sae-connection [sae-connection...]
```

3. (Optional) Specify an SNMP community name for SNMP read-only operations for this VR.

```
[edit shared network device junos_boston virtual-router default]
user@host# set snmp-read-community snmp-read-community
```

4. (Optional) Specify an SNMP community name for SNMP write operations for this virtual router.

```
[edit shared network device junos_boston virtual-router default]
user@host# set snmp-write-community snmp-write-community
```

5. (Optional) Specify service scopes assigned to this virtual router. The scopes are available for subscribers connected to this virtual router for selecting customized versions of services.

```
[edit shared network device junos_boston virtual-router default]
user@host# set scope [scope...]
```

6. (Optional) Specify the plug-ins that track interfaces that the SAE manages on this virtual router.

```
[edit shared network device junos_boston virtual-router default]
user@host# tracking-plug-in [tracking-plug-in...]
```

7. (Optional) Verify your configuration.

```
[edit shared network device junos_boston virtual-router default]
user@host# show
sae-connection 192.168.80.1;
snmp-read-community *****;
snmp-write-community *****;
scope POP-Cambridge;
tracking-plug-in flexRadius;
```

Related Information

For additional information, see the following sources:

- For information about service scopes, see *SRC-PE Services and Policies Guide, Chapter 1, Managing Services with the SRC CLI*
- For information about tracking plug-ins, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Authorization and Accounting Plug-Ins with the CLI*

Configuring the SAE to Manage JUNOS Routing Platforms

A JUNOS routing platform interacts with the SAE by using a JUNOS software process called `sdx`. When the `sdx` process establishes a TCP/IP connection to the SAE, the SAE begins to manage the router. The JUNOS router driver configuration defines parameters related to the interactions between the SAE and the `sdx` process.

Use the following configuration statements to configure the JUNOS router driver:

```
shared sae configuration driver junos {
  beep-server-port beep-server-port;
  tls-beep-server-port tls-beep-server-port;
  connection-attempts connection-attempts;
  keepalive-interval keepalive-interval;
  message-timeout message-timeout;
  batch-size batch-size;
  transaction-batch-time transaction-batch-time;
  sdx-group-name sdx-group-name;
  sdx-session-group-name sdx-session-group-name;
  send-commit-check send-commit-check;
}
```

To configure the JUNOS router driver:

1. From configuration mode, access the configuration statement that configures the JUNOS router driver. In this sample procedure, the JUNOS driver is configured in the west-region group.

```
user@host# edit shared sae group west-region configuration driver junos
```

2. Specify the TCP port number that is used to communicate with the sdx process on JUNOS routing platforms. This port number must match the port number configured in the sdx process on the router.

If you set this value to zero and the TLS BEEP server port is set, the SAE accepts only TLS connections.

```
[edit shared sae group west-region configuration driver junos]
user@host# set beep-server-port beep-server-port
```

3. Specify the TLS port number that is used for TLS connections to the JUNOS routing platform.

If you set this value to zero, the SAE does not accept TLS connections.

```
[edit shared sae group west-region configuration driver junos]
user@host# set tls-beep-server-port tls-beep-server-port
```

4. Specify the number of outstanding connection attempts before new connection attempts are dropped.

```
[edit shared sae group west-region configuration driver junos]
user@host# set connection-attempts connection-attempts
```

5. Specify the interval between keepalive messages sent from the router.

```
[edit shared sae group west-region configuration driver junos]
user@host# set keepalive-interval keepalive-interval
```

6. Specify the amount of time that the router driver waits for a response from the sdx process.

Under a high load the router may not be able to respond fast enough to requests. Change this value only if a high number of timeout events appear in the error log.

```
[edit shared sae group west-region configuration driver junos]
user@host# set message-timeout message-timeout
```

7. Specify the minimum number of service configuration transactions that are committed at the same time

```
[edit shared sae group west-region configuration driver junos]
user@host# set batch-size batch-size
```


8. Specify the maximum time to collect configuration transactions in a batch.

```
[edit shared sae group west-region configuration driver junos]
user@host# set transaction-batch-time transaction-batch-time
```

9. Specify the name of a session group on the JUNOS routing platform in which provisioning objects are stored.

```
[edit shared sae group west-region configuration driver junos]
user@host# set sdx-session-group-name sdx-session-group-name
```

10. Enable or disable commit check. If enabled, a more detailed error message is logged if a batch fails, which lets you verify individual transactions in a batch.

```
[edit shared sae group west-region configuration driver junos]
user@host# set send-commit-check send-commit-check
```

11. (Optional) Verify your configuration.

```
[edit shared sae group west-region configuration driver junos]
user@host# show
beep-server-port 3333;
tls-beep-server-port 0;
connection-attempts 50;
keepalive-interval 45;
message-timeout 30000;
batch-size 10;
transaction-batch-time 2000;
sdx-group-name sdx;
sdx-session-group-name sdx-sessions;
send-commit-check true;
```

Related Information

For additional information, see the following source:

- For information about setting up SAE groups, see *SRC-PE Getting Started Guide, Chapter 16, Setting Up an SAE with the SRC CLI*.

Configuring Secure Connections Between the SAE and JUNOS Routing Platforms

You can use TLS to protect communication between the SAE and JUNOS routing platforms.

To complete the handshaking protocol for the TLS connection, the client (JUNOS routing platform) and the server (SAE) must exchange and verify certificates. You need to create a client certificate and a server certificate. Both certificates must be signed by a certificate authority (CA). JUNOS software supports VeriSign, Inc. (<http://www.verisign.com>). You must then install both certificates on the SAE and on the JUNOS routing platform.

You can use SRC CLI commands to manage certificates manually, or through the Simple Certificate Enrollment Protocol (SCEP).

Certificates are in the format defined in the X.509 standard for public key infrastructure. The certificate requests are in the Public Key Cryptology Standard (PKCS) #10 format.

Tasks to set up the SAE and the JUNOS routing platform to use TLS are:

1. Manually Obtaining Digital Certificates on page 116
- Or
2. Obtaining Digital Certificates through SCEP on page 118
3. Installing the Server Certificate on the Router on page 119
4. Creating a Client Certificate for the Router on page 120
5. Installing the Client Certificate on the Router on page 120
6. Configuring the SAE to Use TLS on page 120
7. Configuring TLS on the SAE on page 120

Manually Obtaining Digital Certificates

You can manually add digital certificates, or you can use SCEP to help manage how you obtain certificates. See *Obtaining Digital Certificates through SCEP* on page 118.

To manually add a signed certificate:

1. Create a certificate signing request.

```
user@host> request security generate-certificate-request subject subject
password password
```

where:

- **subject** is the distinguished name of the SRC host; for example `cn=src1,ou=pop,o=Juniper,l=kanata,st=Ontario,c=Canada`.
- **password** is the password received from the certificate authority for the specified subject.

By default, this request creates the file `/tmp/certreq.csr` and encodes the file by using Privacy-Enhanced Mail (pem) encoding.

2. Copy the file generated in Step 1 to another system, and submit the certificate signing request file generated in Step 1 to VeriSign, Inc. (<http://www.verisign.com>) for signing.

You can transfer the file through FTP by using the **file copy** command.

```
user@host> file copy source_file ftp://username@server[:port]/destination_file
```

VeriSign authenticates you and returns a certificate, signed by them, that authenticates your public key.

3. When you receive the signed certificate, copy the file back to the SRC system to the */tmp* directory.

You can transfer the file through FTP, as shown in Step 2.

4. Add the certificate to the SRC configuration.

```
user@host> request security import-certificate file-name file-name identifier identifier
```

where:

- **file-name** is the name of the certificate file in the */tmp* folder. The file must be in one of the following formats, which is indicated by the following extensions:
 - CER—Windows extension
 - PEM—Privacy-Enhanced Mail encoding
 - DER—Binary encoding
 - BER—Binary encoding
- **identifier** is the name of the certificate.

For example, to import the file **src.cer** that is identified as **web**:

```
user@host> request security import-certificate file-name src.cer identifier web
```

5. Verify that the certificate is part of the SRC configuration.

```
user@host> show security certificate  
web subject:CN=host
```

If there are no certificates on the system, the CLI displays the following message:

```
No entity certificates in key store
```

Obtaining Digital Certificates through SCEP

You can use SCEP to help manage how you obtain digital certificates, or you can manually add certificates. See *Manually Obtaining Digital Certificates* on page 116.

Before you can obtain certificates for your use, you must get the CA's certificate and install it in the local store of trusted certificates.

To add a signed certificate that you obtain through SCEP:

1. Request your CA's certificate through SCEP.

```
user@host> request security get-ca-certificate url url ca-identifier ca-identifier
```

where:

- url is the URL of the certificate authority (which is the SCEP server).
- ca-identifier is the identifier that designates the authority.

For example, to request a certificate from the CA authority SrcCA at a specified URL on the server security_server:

```
user@host> request security get-ca-certificate url
http://security_server:8080/ejbca/publicweb/apply/scep/pkiclient.exe
ca-identifier SrcCA
```

```
Version: 3
Serial Number: 5721058705923989279
Signature Algorithm: SHA1withRSA
Issuer: CN=SrcCA
Valid From: Wed Sep 06 17:00:55 EDT 2006
Valid Until: Sat Sep 03 17:10:55 EDT 2016
Subject: CN=SrcCA
Public key: RSA
Thumbprint Algorithm: SHA1
Thumbprint: 3c 57 a9 77 af 83 3 e9 c7 1e ee e2 4a e8 ff f3 89 f4 11 a9
Do you want to add the above certificate as a trusted CA [yes,no] ? (no) y
```

2. Request that the certificate authority automatically sign the certificate request.

```
user@host> request security enroll subject subject password password
```

where:

- subject is the distinguished name of the SRC host; for example cn=myhost.
- password is the password received from the certificate authority.

For example, to request a certificate from the CA authority SrcCA at a specified URL on the server security_server:

```
user@host> request security enroll url
http://security_server:8080/ejbca/publicweb/apply/scep/pkiclient.exe
identifier web ca-identifier SrcCA subject cn=myhost password mypassword
```

```

Received certificate:
Version: 3
Serial Number: 6822890691617224432
Signature Algorithm: SHA1withRSA
Issuer: CN=SrcCA
Valid From: Tue Sep 19 16:33:11 EDT 2006
Valid Until: Thu Sep 18 16:43:11 EDT 2008
Subject: CN=myhost
Public key: RSA
Do you want to install the above certificate [yes,no] ? (no) y

```

3. Verify that the certificate is part of the SRC configuration.

```

user@host> show security certificate
web subject:CN=myhost

```

If there are no certificates on the system, the CLI displays the following message:

```
No entity certificates in key store
```

Installing the Server Certificate on the Router

The TLS client (JUNOS routing platform) needs a copy of the certificate that was used to sign the SAE certificate so that it can verify the SAE certificate. To install the SAE certificate on the JUNOS routing platform:

1. Include the following statements at the [edit security certificates certificate-authority] hierarchy level.

```

[edit security certificates certificate-authority]
security{
  certificates{
    certificate-authority SAECert{
      file /var/db/certs/cert.pem;
    }
  }
}

```

2. Include the following statements at the [system services service-deployment] hierarchy level.

```

system{
  services{
    service-deployment{
      servers {
        server-address port port-number{
          security-options {
            tls;
          }
        }
      }
    }
  }
}

```

Creating a Client Certificate for the Router

For information about how to obtain a certificate for the router from a certificate authority, see *Obtaining a Certificate from a Certificate Authority* in the *JUNOS System Basics Configuration Guide*.

Installing the Client Certificate on the Router

To install the client (router) certificate on the JUNOS routing platform:

1. Include the following statements at the [edit security certificates certificate-authority] hierarchy level.

```
[edit security certificates certificate-authority]
security{
  certificates{
    local clientCERT { .... } ;
  }
}
```

2. Include the following statements at the [system services service-deployment] hierarchy level.

```
system{
  services{
    service-deployment{
      local-certificate clientCert;
    }
  }
}
```

Configuring the SAE to Use TLS

To configure the SAE to accept TLS connections, enter a port number with the **set beep-server-port** command in the JUNOS router driver configuration.

See *Configuring the SAE to Manage JUNOS Routing Platforms* on page 113.

Configuring TLS on the SAE

Use the following configuration statements to configure TLS on the SAE:

```
shared sae configuration driver junos security {
  need-client-authentication;
  certificate-identifier private-key;
}
```

To configure TLS on the SAE:

1. From configuration mode, access the configuration statement that configures security for the JUNOS TLS connection. In this sample procedure, the JUNOS driver is configured in the west-region group.

user@host# **edit shared sae group west-region configuration driver junos security**

2. (Optional) Specify whether or not the SAE requests a client certificate from the router when a connection to the router is established.

```
[edit shared sae group west-region configuration driver junos security]
user@host# set need-client-authentication
```

3. Specify the name of certificate to be used for TLS communications.

```
[edit shared sae group west-region configuration driver junos security]
user@host# set certificate-identifier private-key
```

4. (Optional) Verify your TLS configuration.

```
[edit shared sae group west-region configuration driver junos security]
user@host# show
need-client-authentication;
certificate-identifier privatekey;
```

Checking Changes to the JUNOS Configuration

The SAE can check the configuration of a JUNOS routing platform under its control to detect whether the configuration has changed by a means other than through the SAE. If the SAE finds a disparity between the router and the SAE configurations, it can take several actions. The SAE checks the configuration installed on the router against the state of the SAE session layer (subscriber, service, and interface sessions). While the check is occurring, the SAE does not handle jobs from the router, and all provisioning activity is blocked, including event notifications.

The SAE can take the following actions if it finds a disparity between the router and SAE configurations:

- The SAE takes the state of the session layer on the router to be correct and updates its local state to be consistent with the router. The SAE then sends stop events for all sessions where the corresponding provisioning in the router has been removed.
- The SAE takes its local state to be the correct state and updates the router to be consistent with its local state.
- The SAE does not solve the state discrepancy. It reports disparities through the SAE device driver event trap called `routerConfOutOfSynch` and through the info log.

Note that it is not possible to check the consistency of individual objects that the SAE provisions. Therefore, modifications to a provisioning object while the SAE is disconnected from the router cannot be detected.

Setting Up Periodic Configuration Checking

Use the following configuration statements to configure the SAE to periodically check the configuration of the JUNOS routing platform:

```
shared sae configuration driver junos configuration-checking
  configuration-checking-schedule configuration-checking-schedule;
  configuration-checking-action (enforce | synchronize | detect);
}
```

To configure the SAE to periodically check the configuration of the JUNOS routing platform:

1. From configuration mode, access the configuration statement that configures the configuration checking feature.

```
user@host# edit shared sae configuration driver junos configuration-checking
```

2. Specify when the SAE checks the router configuration.

```
[edit shared sae configuration driver junos configuration-checking]
user@host# set configuration-checking-schedule configuration-checking-schedule
```

3. Specify the action that the SAE takes when it detects disparities between the configuration of the SAE and the configuration on the router.

```
[edit shared sae configuration driver junos configuration-checking]
user@host# set configuration-checking-action enforce | synchronize | detect
```

4. (Optional) From operational mode, verify your configuration checking configuration.

```
[edit shared sae configuration driver junos configuration-checking]
user@host# show
configuration-checking-schedule "0 0 * * * * *";
configuration-checking-action synchronize;
```

Using SNMP to Retrieve Information from JUNOS Routing Platforms

You can use SNMP to retrieve information from the router. For example, if you create a router initialization script that uses SNMP, you need to specify the SNMP communities that are on the router.

We recommend that you specify SNMP communities for each virtual router. (See *Adding Virtual Routers Individually* on page 112.) You can also configure global default SNMP communities.

Configuring Global SNMP Communities in the SRC Software

You can configure global default SNMP communities that are used if a VR does not exist on the router or if the community strings have not been configured for the VR.

Use the following configuration statements to configure global default SNMP communities:

```
shared sae configuration driver snmp {
    read-only-community-string read-only-community-string;
    read-write-community-string read-write-community-string;
}
```

To configure global default SNMP communities:

1. From configuration mode, access the configuration statements that configure default SNMP communities.

```
user@host# edit shared sae configuration driver snmp
```

2. Configure the default SNMP community string used for read access to the router.

```
[edit shared sae configuration driver snmp]
user@host# set read-only-community-string read-only-community-string
```

3. Configure the default SNMP community string used for write access to the router.

```
[edit shared sae configuration driver snmp]
user@host# set read-write-community-string read-write-community-string
```

4. (Optional) Verify your configuration.

```
[edit shared sae configuration driver snmp]
user@host# show
read-only-community-string *****;
read-write-community-string *****;
```

Developing Router Initialization Scripts

When the SAE establishes a connection with a router, it can run a router initialization script to customize the setup of the connection. Router initialization scripts are run when the connection between a router and the SAE is established and again when the connection is dropped.

We provide the `iorPublisher` script in the `/opt/UMC/sae/lib` folder. The `iorPublisher` script publishes the IOR of the SAE into an internal part of the shared configuration so that a NIC can associate a router with an SAE.

Interface Object Fields

Router initialization scripts are written in the Python programming language (www.python.org) and executed in the Jython environment (www.jython.org).

Router initialization scripts interact with the SAE through an interface object called `Ssp`. The SAE exports a number of fields through the interface object to the script and expects the script to provide the entry point to the SAE.

Table 8 describes the fields that the SAE exports.

Table 8: Exported Fields

Ssp Attribute	Description
<code>Ssp.properties</code>	System properties object (class: <code>java.util.Properties</code>)—The properties should be treated as read-only by the script.
<code>Ssp.errorLog</code>	Error logger—Use the <code>SsperrorLog.println (message)</code> to send error messages to the log.
<code>Ssp.infoLog</code>	Info logger—Use the <code>Ssp.infoLog.println (message)</code> to send informational messages to the log.
<code>Ssp.debugLog</code>	Debug logger—Use the <code>Ssp.debugLog.println (message)</code> to send debug messages to the log.

The router initialization script must set the field `Ssp.routerInit` to a factory function that instantiates a router initialization object:

- `<VRName>` —Name of the virtual router in which the COPS client has been configured, in the format: `virtualRouterName@RouterName`
- `<virtuallp>` —Virtual IP address of the SAE (string, dotted decimal; for example: `192.168.254.1`)
- `<reallp>` —Real IP address of the SAE (string, dotted decimal; for example, `192.168.1.20`)
- `<VRlp>` —IP address of the virtual router (string, dotted decimal)
- `<transportVR>` —Name of the virtual router used for routing the COPS connection, or `None`, if the COPS client is directly connected

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
               virtuallp,
               reallp,
               VRlp,
               transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a given COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a COPS server connection is established. In case of problems, an exception should be raised that leads to the termination of the COPS connection.

Required Methods

Instances of the interface object must implement the following methods:

- *setup()*—Is called when the COPS server connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the COPS connection.
- *shutdown()*—Is called when the COPS server connection is terminated to the virtual router. This method should not raise any exceptions in case of problems.

Example: Router Initialization Script

The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality; it just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.println("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.println("Setup connection to VR %(vrName)s" %
                           vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.println("Shutdown connection to VR %(vrName)s" %
                           vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

Specifying Router Initialization Scripts on the SAE

Use the following configuration statements to specify router initialization scripts for JUNOS routing platforms:

```
shared sae configuration driver scripts {
    extension-path extension-path;
    general general;
    junos junos;
}
```

To configure router initialization scripts for JUNOS routing platforms:

1. From configuration mode, access the configuration statements that configure router initialization scripts. In this sample procedure, the scripts are configured in the west-region group.

```
user@host# edit shared sae group west-region configuration driver scripts
```

2. Specify the router initialization script for JUNOS routing platforms.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set junos junos
```

3. Configure a router initialization script that can be used for all types of routers that the SRC software supports.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set general general
```

4. Configure a path to router initialization scripts that are not in the default location, */opt/UMC/sae/lib*.

```
[edit shared sae group west-region configuration driver scripts]
user@host# set extension-path extension-path
```

5. (Optional) From operational mode, verify your router initialization script configuration.

```
[edit shared sae group west-region configuration driver scripts]
user@host# show
extension-path ;
junos iorPublisher;
```

Accessing the Router CLI

You can access the CLIs of Juniper Networks routers through a Telnet or secure shell connection.

- To open a Telnet session to a router, use the **telnet** operational mode command. For example:

```
user@host> telnet 10.10.10.3
```

- To open a secure shell connection, use the **ssh** operational command. For example:

```
user@host> ssh host 10.10.10.3
```

Configuring JUNOS Routing Platforms to Interact with the SAE

To configure the JUNOS routing platform to interact with the SAE:

1. Include the following statements at the [edit system services service-deployment] hierarchy level.

```
[edit system services service-deployment]
servers server-address {
  port port-number;
}
source-address source-address;
```

2. Use the following guidelines for the variables in these statements.

server-address

- Specifies the IP address of the host on which you install the SAE.
- Value—IP address
- Guidelines—Be sure this setting matches the corresponding value in the SAE configuration.
- Default—None
- Example—192.0.2.2

port-number

- Specifies the port number for the SAE.
- Value—TCP port number
- Guidelines—Be sure this setting matches the corresponding value in the SAE configuration.
- Default—3333
- Example—3333

source-address

- Specifies the IP address of the source that sends traffic to the SAE.
- Value—IP address
- Guidelines—This setting is optional.
- Default—None
- Example—192.0.2.2

Configuring the JUNOS Routing Platform to Apply Changes It Receives from the SAE

To configure the JUNOS routing platform to receive configuration statements from the SAE and apply those statements to the configuration:

1. Create a configuration group called `sdx` that contains the configuration statements that the SAE sends to the JUNOS routing platform. To do so, include the `groups` statement at the `[edit]` level, and specify the name `sdx`.

```
[edit]
groups {
  sdx;
}
```

2. Configure the JUNOS routing platform to apply these statements to the configuration. To do so, include the `apply-groups` statement at the `[edit]` level.

```
[edit]
set apply-groups sdx;
```

Disabling Interactions Between the SAE and JUNOS Routing Platforms

To disable the SRC software process, enter the following command:

```
root@ui1#set system processes service-deployment disable
root@ui1#commit
```

When you disable the SRC software process, it is still available on the JUNOS routing platform.

To reenable the SRC software process, enter the following command:

```
root@ui1#delete system processes service-deployment disable
root@ui1#commit
```

The SRC software process attempts to reconnect the JUNOS routing platform to the SAE.

Monitoring Interactions Between the SAE and JUNOS Routing Platforms

Use the following command on JUNOS routing platforms to monitor the connection between the JUNOS routing platform and the SAE.

```
root@ui1> show system services service-deployment
Connected to 172.17.20.151 port 3333 since 2004-02-06 14:50:31 PST
Keepalive settings: Interval 15 seconds
Keepalives sent: 100, Last sent: 6 seconds ago
Notifications sent: 0
Last update from peer: 00:00:06 ago
```

You can also monitor the interactions between the SRC software and JUNOS routing platforms in the log files for the SAE and in the log files generated by the SRC software process on the JUNOS routing platform.

- For information about configuring logging for the SAE, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 3, Configuring Logging for SRC Components with the CLI*.
- For information about configuring logging on JUNOS routing platforms, see *JUNOS System Basics Configuration Guide*.

Troubleshooting Problems with the SRC Software Process

To troubleshoot SRC problems on the JUNOS routing platform, review the log files for the SAE and the log files generated by the SRC software process on the router. If the log files indicate that the SRC software process on the JUNOS routing platform is not responding:

1. Look at the status of the process on the JUNOS routing platform.

```
root@ui1>show system services service-deployment
Connected to 172.17.20.151 port 3333 since 2004-02-06 14:50:31 PST
Keepalive settings: Interval 15 seconds
Keepalives sent: 100, Last sent: 6 seconds ago
Notifications sent: 0
Last update from peer: 00:00:06 ago
```

2. If you see the message “error: the service-deployment subsystem is not running,” reenable the SRC software process. See *Disabling Interactions Between the SAE and JUNOS Routing Platforms* on page 128.
3. If the process is already enabled, review the configurations of the router and the SAE in the directory, and fix any problems.
4. Restart the SRC software process on the router.

```
root@ui1>restart service-deployment
```

The SAE synchronizes with the SRC software process and deletes unnecessary data from the router.

Deleting All SRC Data on JUNOS Routing Platforms

If deleting parts of the SRC data on a JUNOS routing platform fails to solve problems, delete all the SRC data and restart the SRC software process. To do so:

1. Delete all SRC interfaces and services.

```
delete groups sdx
root@ui1#commit
```

2. If you are running SDX software releases 5.0 through 6.1, you should also delete interface sessions. (After release 6.2, session data is no longer stored on the router, it is stored on the SAE host using the session store feature.)

```
delete groups sdx-sessions
root@ui1#commit
```

3. Restart the SRC software process on the router.

```
root@ui1>restart service-deployment
```

Viewing the State of JUNOS Device Drivers with the SRC CLI

To display the state of JUNOS drivers, use the following operational mode command:

```
show sae drivers <device-name device-name> < (brief) > <maximum-results maximum-results>
```

For example:

```
user@host> show sae drivers device-name default@jrouter
JUNOS Driver
Device name                default@jrouter
Device type                junos
Device IP                  /10.10.6.113:1879
Local IP                   10.10.6.113
TransportRouter
Device version             8.2R1.7
Start time                 Thu Mar 08 21:00:50 UTC 2007
Number of notifications    0
Number of processed added  0
Number of processed changed 0
Number of processed deleted 0
Number of provisioning attempt 0
Number of provisioning attempt failed 0
Device type                JunosRouterDriver
Job queue size             0
Number of SAP              3
Number of PAP              0
Start time                 Thu Mar 08 21:00:55 UTC 2007
End time                   Thu Mar 08 21:00:55 UTC 2007

Transaction Manager
Transaction queue size 0
Router name              default@troll
```

Viewing Statistics for Specific JUNOS Device Drivers with the SRC CLI

To display statistics for a specific JUNOS device driver, use the following operational mode command:

```
show sae statistics device <name name> < (brief) >
```

For example:

```
user@host> show sae statistics device name default@jrouter
SNMP Statistics
Add notification handle time 7
Change notification handle time 0
Client ID                    default@troll
Delete notification handle time 0
Failover IP                  0.0.0.0
Failover port                0
Handle message time         40
Job queue age               0
Job queue time              0
Number message send         3
Number of added jobs        0
Number of add notifications 0
Number of change notifications 0
Number of delete notifications 0
```


Number of managed interfaces	3
Number of message errors	0
Number of message timeouts	0
Number of removed jobs	0
Number of user session established	0
Number of user session removed	0
Router type	JUNOS
Up time	7036120
Using failover server	false

Viewing Statistics for All JUNOS Device Drivers with the SRC CLI

To display SNMP statistics for all JUNOS device drivers, use the following operational mode command:

```
show sae statistics device common junos
```

For example:

```
user@host> show sae statistics device common junos
SNMP Statistics
Driver type                JUNOS
Number of close requests   0
Number of connections accepted 0
Number of current connections 0
Number of open requests    0
Server address             0.0.0.0
Server port                3288
Time since last redirect    0
```

Viewing the State of JUNOS Device Drivers with the C-Web Interface

If the log files indicate a problem with a specific driver, review the configuration of the associated JUNOS router driver with C-Web.

1. Select **SAE** from the side pane, and click **Drivers**.

The Drivers pane appears.

The screenshot shows the Juniper C-Web interface. On the left is a sidebar with a tree view containing: Monitor, ACP, CLI, Component, Date, Disk, Interfaces..., JPS, NIC, NTP, Redirect Server, Route..., SAE, Security, and System. The 'SAE' item is selected, and the 'Drivers' sub-pane is active. The main content area has a title bar 'SAE Drivers' and a breadcrumb 'SAE > Drivers'. Below the title bar are three input fields: 'Name Of Device Driver' (a text box), 'Style' (a dropdown menu), and 'Maximum Results' (a text box). To the right of these fields are three informational text blocks: 'Name of device drivers. Please enter: All or part of the device driver name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.', 'Output style Choices: brief: Display only virtual router names', and 'Number of results to be displayed. Legal range: 1 .. INF Default value: 25'. Below the input fields are 'OK' and 'Reset' buttons. At the bottom of the interface is a footer with copyright information and the Juniper logo.

2. In the Name of Device Driver box, enter a full or partial device driver name for which you want to display information, or leave the box blank to display all devices. Use the format:

default@<router name>

3. Select an output style from the Style list.
4. In the Maximum Results box, enter the maximum number of results that you want to receive.
5. Click **OK**.

The Drivers pane displays information about the JUNOS device driver.

Viewing Statistics for Specific JUNOS Device Drivers with the C-Web Interface

To view SNMP statistics about devices:

1. Select **SAE** from the side pane, click **Statistics**, and then click **Device**.

The Device pane appears.

The screenshot shows the Juniper C-Web interface. On the left is a navigation pane with categories like Monitor, ACP, CLI, Component, Date, Disk, Interfaces..., JPS, NIC, NTP, Redirect Server, Route..., SAE, Security, and System. The main area is titled 'SAE' and 'Device'. It contains a 'Device Name' text input field, a 'Style' dropdown menu, and 'OK' and 'Reset' buttons. To the right of the input fields is a help text box that reads: 'Name of a device. Please enter: All or part of the device name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.' Below this, it says 'Output style Choices: brief: Display only device names'. At the bottom of the interface, there is a copyright notice: 'Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice, Privacy.' and the Juniper logo with the tagline 'Juniper Your Net.'

2. In the Device Name box, enter a full or partial device name for which you want to display information, or leave the box blank to display all devices.
3. Select an output style from the Style list.
4. Click **OK**.

The Device pane displays statistics for all devices.

Viewing Statistics for All JUNOS Device Drivers with the C-Web Interface

To view SNMP statistics about specific devices:

1. Select **SAE** from the side pane, click **Statistics**, click **Device**, and then click **Common**.

The Common pane appears.

The screenshot shows the Juniper C-Web Interface. On the left is a sidebar with a 'Monitor' section containing links like ACP, CLI, Component, Date, Disk, Interfaces..., JPS, NIC, NTP, Redirect Server, Route..., SAE, Security, and System. The main area is titled 'SAE' and 'Common'. It contains a 'Device Name' text box and a 'Type' dropdown menu. To the right of these fields is a text area with instructions: 'Name of a device. Please enter: All or part of the device name. For JUNOS router drivers and PCMM drivers, use the format default@routerName.' Below this is another instruction: 'Display SNMP statistics for a specified device driver type. Choices: junos: Display SNMP statistics for JUNOS router drivers; junose-cops: Display SNMP statistics for JUNOSe router drivers; packetoable-cops: Display SNMP statistics for PCMM device drivers; proxy: Display SNMP statistics for third-party drivers'. At the bottom of the form are 'OK' and 'Reset' buttons. The top of the interface shows 'Logged in as: admin' and links for 'About', 'Refresh', and 'Logout'. The bottom footer contains copyright information for Juniper Networks, Inc. 2007 and the Juniper logo.

2. In the Device Name box, enter a full or partial device name for which you want to display information, or leave the box blank to display all devices.
3. Select the **junos** from the Type list:
4. Click **OK**.

The Common pane displays statistics for the specified device.

Chapter 8

Using JUNOS Routing Platforms in the SRC Network with a Solaris Platform

This chapter describes how to set up the SRC software on a Solaris platform with the SRC configuration applications that run only on Solaris platforms. It also shows how to set up JUNOS routing platforms so that the routing platforms can be used the SRC network. It includes information about how to monitor the interactions between the SAE and JUNOS routing platforms and how to troubleshoot SRC problems on JUNOS routing platforms.

You can also use the CLI that runs on Solaris platforms and the C-series platform to configure the SRC system to work with JUNOS routing platforms. See *Chapter 7, Using JUNOS Routing Platforms in the SRC Network with the SRC CLI*.

Topics in this chapter include:

- BEEP Connection Between JUNOS Routing Platforms and the SAE on page 136
- Adding JUNOS Routing Platforms and Virtual Routers on page 136
- Configuring the SAE to Manage JUNOS Routing Platforms on page 144
- Configuring Secure Connections Between the SAE and JUNOS Routing Platforms on page 148
- Checking Changes to the JUNOS Configuration on page 153
- Using SNMP to Retrieve Information from JUNOS Routing Platforms on page 156
- Developing Router Initialization Scripts on page 157
- Specifying Router Initialization Scripts on the SAE on page 159
- Accessing the Router CLI on page 160
- Configuring JUNOS Routing Platforms to Interact with the SAE on page 162
- Disabling Interactions Between the SAE and JUNOS Routing Platforms on page 163

- Monitoring Interactions Between the SAE and JUNOS Routing Platforms on page 164
- Troubleshooting SRC Problems on JUNOS Routing Platforms on page 164

BEEP Connection Between JUNOS Routing Platforms and the SAE

For information about which JUNOS routing platforms and releases a particular SRC release supports, see the SRC *Release Notes*.

The SAE interacts with a JUNOS software process, referred to as the SRC software process in this documentation, on the JUNOS routing platform. The SAE and the SRC software process communicate using the Blocks Extensible Exchange Protocol (BEEP). You can secure the BEEP connection by using Transport Layer Security (TLS).

When the SRC software process establishes a BEEP session for the SAE, the SAE configures an interface on the JUNOS routing platform. The SAE builds the configuration for an interface using the policies stored in the directory. If the policies are subsequently modified, the SAE builds a new configuration and reconfigures the interface on the JUNOS routing platform. The JUNOS routing platform stores data about interfaces and services that the SAE manages in a configuration group called *sdx*. You must create this configuration group on the JUNOS routing platform.

Adding JUNOS Routing Platforms and Virtual Routers

On JUNOS routing platforms, the SAE manages interfaces. The SRC software associates a virtual router called default with each JUNOS routing platform. Each JUNOS routing platform in the SRC network and its associated virtual router (VR) called default must appear in the directory. The VRs are not actually configured on the JUNOS routing platform; the VR in the directory provides a way for the SAE to manage the interfaces on the JUNOS routing platform.

There are two ways to add routers to the directory:

- Use SDX Admin to detect operative routers in the SRC network and add them to the directory. This operation creates a VR called default in the directory for each detected JUNOS routing platform.
- Add each router and VR individually. You need to add routers and VRs individually if you use an LDAP client other than SDX Admin or if you want to add inoperative routers.



NOTE: You must define connected SAEs for each router in the virtual router object of the directory. This step is required for the SAE to work with the router. See *Specifying the SAEs That Can Manage the Router* on page 143.

Adding Operative JUNOS Routing Platforms

To add routers that are currently operative and have an operating SNMP agent:

1. In the SDX Admin navigation pane, select **o = Network**, and right-click.
2. Select **Discover Network**.

The Discover Network dialog box appears.

3. Enter the IP address, the prefix of the network, and the SNMP community string.
4. Click **OK**.

For each JUNOS routing platform, the software creates one VR called default. You can modify the configuration of these objects. For information about configuring these objects, see *Adding Routers Individually* on page 137 and *Adding Virtual Routers Individually* on page 139.

Adding Routers Individually

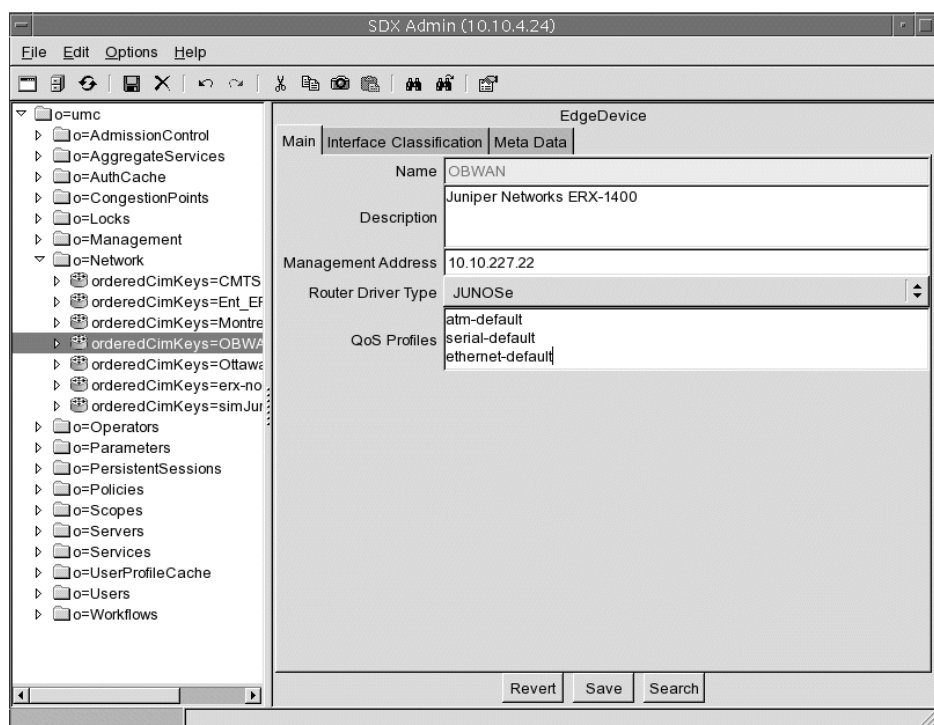
To add a single router with SDX Admin:

1. In the navigation pane, right-click the Network folder, and select **New > EdgeDevice**.

The New EdgeDevice dialog box appears.

2. Enter the name of the router exactly as it is configured in the JUNOS software, and click **OK**.

The new device appears in the navigation pane, and the Main tab of the EdgeDevice pane appears.



3. Edit or accept the default values for the router fields.

See *Router Fields* on page 138.

4. Click **Save**.

Router Fields

In SDX Admin, you can modify the following fields in the content pane for a router (*orderedCimKeys = < EdgeDeviceName >, o = network, o = umc*).

Description

- Information about this device; keywords that the SRC find utility uses.
- Value—Text string
- Example—ERX-1400 router located in Ottawa

Management Address

- IP address of the router or CMTS device. If you add a router using the discover network feature, the software automatically adds the IP address of the first SNMP agent on the router to respond to the discover request.
- Value—IP address
- Example—192.0.1.1

Router Driver Type

- Type of device that this directory object will be used to manage.
- Value
 - JUNOSe—JUNOSe router
 - JUNOS—JUNOS routing platform
 - PCMM—CMTS device
- Default—No value

QoS Profiles

- For JUNOSe routers, specifies quality of service (QoS) profiles that are configured on the router.
- Value—List of QoS profiles on separate lines
- Guideline—This field applies to JUNOSe routers only
- Example—atm-default

Adding Virtual Routers Individually

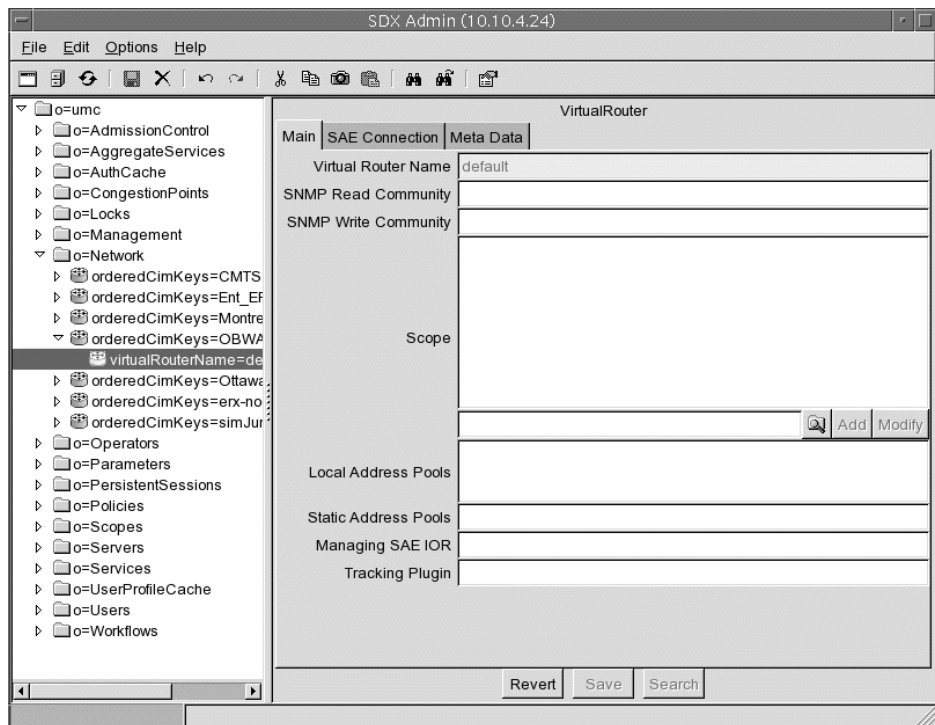
To add a VR with SDX Admin:

1. In the navigation pane, right-click the device to which you want to add the VR, and select **New > VirtualRouter**.

The New VirtualRouter dialog box appears.

2. Enter the name of the VR, and click **OK**.
 - For JUNOSe routers, the name of the VR, which is case sensitive, must exactly match the name of the VR configured on the router.
 - For JUNOS routing platforms and CMTS devices, use the name default.

The new VR appears in the navigation pane, and the Main tab of the VirtualRouter pane appears.



3. Enter or accept the default values for the virtual router fields.

See *Virtual Router Fields* on page 140.

4. Select the **SAE Connection** tab in the VirtualRouter pane, and add SAEs that are connected to the router. See *Specifying the SAEs That Can Manage the Router* on page 143.



NOTE: This step is required for the SAE to work with the router.

5. Click **Save**.

Virtual Router Fields

In SDX Admin, you can modify the following fields in the content pane for a virtual router (*virtualRouterName* = < *virtualRouterName* > , *orderedCimKeys* = < *EdgeDeviceName* > , *o* = *network*, *o* = *umc*).

SNMP Read Community

- SNMP community name associated with SNMP read-only operations for this VR.
- Value—Text string
- Example—admin

SNMP Write Community

- SNMP community name associated with SNMP write operations for this VR.
- Value—Text string
- Example—public

Scope

- Service scopes assigned to this VR.
- Value—Text string
- Example—POP-Westford

Local Address Pools

- List of IP address pools that a JUNOS VR currently manages and stores.
- Value—You can specify an unlimited number of ranges of local IP address pools for JUNOS VRs. You can specify either the first and last addresses in a range or the first IP address and a factor that indicates the start of the range. You can also specify IP addresses to exclude. Use spaces in the syntax only to separate the first and last explicit IP addresses in a range.

The IP pool syntax has the format:

```
([<ipAddressStart> <ipAddressEnd>] |
{<ipBaseAddress>/(<mask> | <digitNumber>)(,<ipAddressExclude>)*})
```

where:

- <ipAddressStart> —First IP address (version 4 or 6) in a range
- <ipAddressEnd> —Last IP address (version 4 or 6) in a range
- <ipBaseAddress> —Network base address
- <mask> —IP address mask
- <digitNumber> —Integer specifying the number of significant digits of the first IP address in the range
- <ipAddressExclude> —List of IP addresses to be excluded from the range
- |—Choice of expression; choose either the expression to the left or the expression to the right of this symbol
- *—Zero or more instances of the preceding group
- Guidelines—Configure this field on JUNOS VRs only. If you do not configure the **PoolPublisher** router initialization scripts for a JUNOS router, configure this field for the JUNOS VR.
- Default—No value

- Example—This example shows four ranges for the IP address pool.

```
([10.10.10.5 10.10.10.250]
{10.20.20.0/24}
{10.21.0.0/255.255.0.0}
{10.20.30.0/24,10.20.30.1})
```
- The first range (a simple range) specifies all the IP addresses between the two IP addresses 10.10.10.5 and 10.10.10.250.
- The second range specifies all the IP addresses in the range 10.20.20.0 to 10.20.20.255.
- The third range uses a network mask to specify all the IP addresses in the range 10.21.0.0 to 10.21.255.255.
- The fourth range specifies all the addresses of the network 10.20.30.0 to 10.20.30.255, excluding the address 10.20.30.1.

Static Address Pools

- List of IP address pools that a JUNOS VR manages but does not store. You can configure these address pools only in the SRC software.
- Value—See the field Local Address Pools.
- Guidelines—Configure this field on JUNOS and CMTS VRs only.
- Default—No value
- Example—([10.10.10.5 10.10.10.250] { 10.20.20.0/24})

Managing SAE IOR

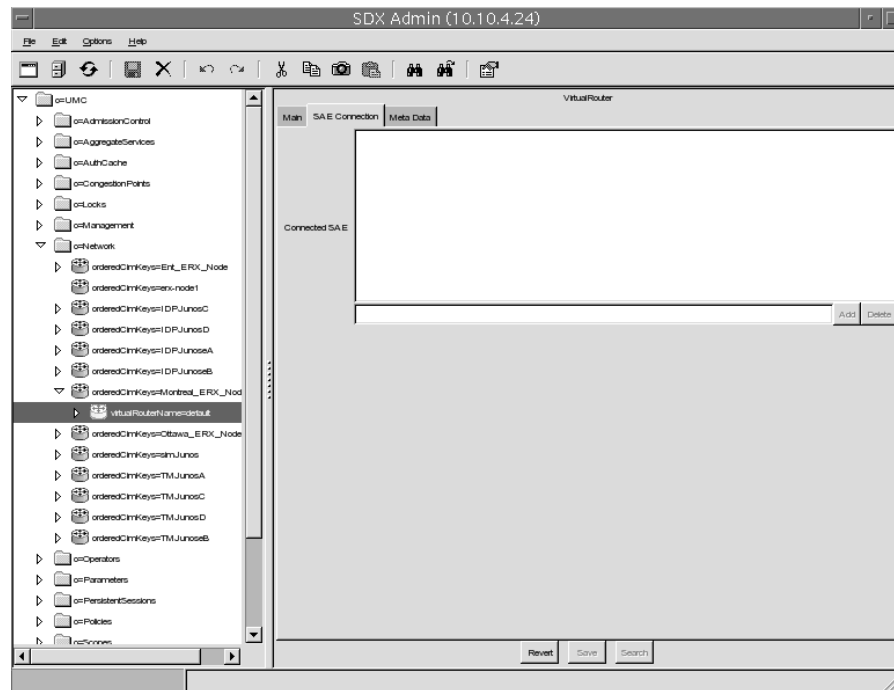
- Common Object Request Broker Architecture (CORBA) reference for the SAE managing this VR.
- Value—One of the following items:
 - The actual CORBA reference for the SAE
 - The absolute path to the interoperable object reference (IOR) file
 - A corbaloc URL in the form corbaloc::<host>:8801/SAE
 - <host> is the name or IP address of the SAE host.
- Default—No value
- Guidelines—The **PoolPublisher** and **IorPublisher** router initialization scripts provide this information when the router connects to the SAE. If you do not select one of these router initialization scripts, enter a value in this field.
- Example—One of the following items:
 - Absolute path—`/opt/UMC/sae/var/run/sae.ior`
 - corbaloc URL—`corbaloc::boston:8801/SAE`
 - Actual IOR—`IOR:0000000000000002438444C3A736D67742E6A756E697...`

Tracking Plug-in

- Plug-ins that track interfaces that the SAE manages on this VR. The SAE calls these plug-in instances for every interface it manages. The SAE calls these plug-ins after an interface comes up, when new policies are installed on the interface, and when the interface goes down.
- Value—Comma-separated list of plug-in instances
- Guidelines—Enter plug-in instances and network information collector (NIC) SAE plug-in agents that are specific to this VR.
- Default—No value
- Example—nicsae, flexRadius

Specifying the SAEs That Can Manage the Router

You must add the addresses of SAEs that can manage this router. This step is required for the SAE to work with the router. To add the SAEs, select the SAE Connection tab in the VirtualRouter pane.



Adding an SAE

To add an SAE:

1. Type the IP address of the SAE in the field below the Connected SAE box.
2. Click **Add**.

Modifying an SAE Address

To modify an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Modify the IP address in the field below the Connected SAE box.
3. Click **Modify**.

Deleting an SAE Address

To delete an SAE address:

1. Click the IP address of the SAE in the Connected SAE box.
2. Remove the IP address from the field below the Connected SAE box.
3. Click **Delete**.

Connected SAE

- SAEs that are connected to the router or CMTS device.
- Value—IP addresses
- Default—No value

Configuring the SAE to Manage JUNOS Routing Platforms

A JUNOS routing platform interacts with the SAE by using a JUNOS software process called `sdx`. When the `sdx` process establishes a TCP/IP connection to the SAE, the SAE begins to manage the router. The JUNOS router driver configuration defines parameters related to the interactions between the SAE and the `sdx` process.

To use SDX Configuration Editor to configure a JUNOS router driver:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the Router tab, and expand the **JUNOS Router Driver** section.

JUNOS Router Driver	
BEEP Server Port	3333
TLS BEEP Server Port	3434
Connection Attempts	50
Keepalive Interval [s]	45
Message Timeout [ms]	30000
Batch Size	10
Transaction Batch Time [ms]	2000
SDX Group Name	sdx
SDX Session Group Name	sdx-sessions
Send Commit Check	true

3. Edit or accept the default values in the fields.

See *JUNOS Router Driver Fields* on page 145.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

JUNOS Router Driver Fields

In SDX Configuration Editor, you can edit the following fields in the JUNOS Router Driver section of the Router pane in an SAE configuration file.

BEEP Server Port

- TCP port number that is used to communicate with the sdx process on JUNOS routing platforms. This port number must match the port number configured in the sdx process on the router.
- Value—TCP port number; if this value is set to zero and the TLS BEEP server port is set, only TLS connections will be accepted.
- Guidelines—If you change this port number, you need to restart the SAE before the change takes effect.
- Default—3333
- Property name—Router.junos.server_port

TLS BEEP Server Port

- TLS port number that is used for TLS connections to the JUNOS routing platform.
See *Configuring Secure Connections Between the SAE and JUNOS Routing Platforms* on page 148.
- Value—TLS port number. If the number is set to 0, the SAE does not accept TLS connections.
- Guidelines—If you change this port number, you need to restart the SAE before the change takes effect.
- Default—3434
- Property name—Router.junos.server_tls_port

Connection Attempts

- Number of socket connection attempts that are accepted while the SAE creates sockets before new attempts are dropped.
- Value—Positive value greater than 0; if the value is equal to or less than 0, the default value is used
- Default—50
- Property name—Router.junos.backlog_connections

Keepalive Interval [s]

- Interval between keepalive messages sent from the router. The sdx process on the router monitors the connection to the SAE by sending keepalive messages at one-third the specified interval. If the sdx process does not receive the expected keepalive answer within the specified timeout, it closes the connection. A short interval results in a high load on the BEEP interface. A long interval results in a long time before a connection failure is detected.
- Value—Number of seconds in the range 0–2147483647. A value of 0 means that timeout is disabled.
- Default—45
- Property name—Router.junos.keepalive

Message Timeout [ms]

- Amount of time that the router driver waits for a response from the sdx process. Under a high load the router may not be able to respond fast enough to requests. Change this value only if a high number of timeout events appear in the error log.
- Value—Number of milliseconds in the range 0–2147483647
- Default—30000
- Property name—Router.junos.message_timeout

Batch Size

- Minimum number of service configuration transactions that are committed at the same time. If any of the transactions in a batch fails, all transactions are aborted, and the associated service activations or deactivations fail.
- Value—Integer in the range 0–2147483647
- Guidelines—To control maximum latency for a job when services are activated in parallel, specify 120 % of the number of CORBA threads as the batch size.
- Default—10
- Property name—Router.junos.batch_size

Transaction Batch Time [ms]

- Maximum time to collect configuration transactions in a batch. The batch is completed if either the batch size or the batch time is reached.
- Value—Number of milliseconds in the range 0–2147483647
- Guidelines—The completion time is calculated from the creation of a batch. Note that the batch time is a function of the total configuration size and not of the number of commands in the configuration transactions.
- Default—2000
- Property name—Router.junos.batch_time

SDX Group Name

- Name of group on the JUNOS routing platform in which provisioning objects are stored.
- Value—Name configured on the JUNOS routing platform
- Default—sdx
- Property name—Router.junos.group.config

SDX Session Group Name

- Name of group on the JUNOS routing platform in which session objects are stored.
- Value—Name configured on the JUNOS routing platform
- Default—sdx-sessions
- Property name—Router.junos.group.session

Send Commit Check

- Enables or disables commit check. If enabled, a more detailed error message is logged if a batch fails, which lets you verify individual transactions in a batch.
- Value—True or false
- Guidelines—To maximize service activation performance, commit check should be disabled.
- Default—True
- Property name—Router.junos.send_commit_check

Configuring Secure Connections Between the SAE and JUNOS Routing Platforms

You can use TLS to protect communication between the SAE and JUNOS routing platforms.

To complete the handshaking protocol for the TLS connection, the client (JUNOS routing platform) and the server (SAE) must exchange and verify certificates. You need to create a client certificate and a server certificate. Both certificates must be signed by a certificate authority (CA). JUNOS software supports VeriSign, Inc. (<http://www.verisign.com>). You must then install both certificates on the SAE and on the JUNOS routing platform.

To set up the SAE and the JUNOS routing platform to use TLS, perform the following tasks:

1. Creating a Server Certificate for the SAE on page 148
2. Installing the Server Certificate on the SAE on page 149
3. Installing the Server Certificate on the Router on page 150
4. Creating a Client Certificate for the Router on page 150
5. Installing the Client Certificate on the Router on page 150
6. Installing the Client Certificate on the SAE on page 151
7. Configuring the SAE to Use TLS on page 151
8. Configuring the Keystore for TLS Certificates and Keys on page 151

Creating a Server Certificate for the SAE

The SRC software provides a sample security certificate that you must replace with a real one. You can obtain a signed certificate from a CA. The SAE stores certificates in a keystore, which is a database of keys and certificates from trusted entities.

To remove the sample certificate and create a site certificate:

1. Access the SAE installation directory.

```
cd /opt/UMC/sae
```

2. Remove the sample certificate.

```
rm -f lib/jetty/saeKeystore
```

3. Generate a self-signed certificate using the **keytool** command; for example:

```
/opt/UMC/jre/bin/keytool -genkey -keyalg RSA -keystore  
keystore/keystore.jks -keypass router -storepass router -alias sae -dname  
<DN> -validity 365
```

The values specified for the **-keystore**, **-keypass**, **-storepass**, and **-alias** arguments must match the following values that you configure for the keystore on the SAE:

- The value of the **-keystore** argument must match the value of the Keystore Location field.
- The value of the **-keypass** and **-storepass** arguments must both match the value of the Keystore Password field.

See *Configuring the Keystore for TLS Certificates and Keys* on page 151.

Replace `<DN>` with the distinguished name that identifies your HTTPS server. For example, if XYM Corp in Canada has an HTTPS server with a hostname of `ssp1.domain.org`, then the DN might be:

```
"cn=ssp1.domain.org, o=XYM Corp, c=CA"
```

Be sure to include the quotation marks. Do not use the `#` character in DNs.

For complete documentation of the Java **keytool**, see:

<http://java.sun.com/j2se/1.4.1/docs/tooldocs/solaris/keytool.html>

4. Create a certificate signing request (CSR).

```
/opt/UMC/jre/bin/keytool -certreq -alias sae -file server.csr -keypass router  
-keystore keystore/keystore.jks -storepass router
```

The command creates a CSR and places it in the *server.csr* file.

5. Send the CSR from the file */opt/UMC/sae/server.csr* for signing to VeriSign, Inc. (<http://www.verisign.com>).

VeriSign authenticates you and returns a certificate, signed by them, that authenticates your public key.

Installing the Server Certificate on the SAE

To install the server certificate on the SAE, import the server certificate into the SAE keystore using the **keytool** command:

```
/opt/UMC/jre/bin/keytool -import -alias sae -file server.crt -keypass router  
-noprompt -trustcacerts -keystore keystore/keystore.jks -storepass router
```

Installing the Server Certificate on the Router

The TLS client (JUNOS routing platform) needs a copy of the certificate that was used to sign the SAE certificate so that it can verify the SAE certificate. To install the SAE certificate on the JUNOS routing platform:

1. Include the following statements at the [edit security certificates certificate-authority] hierarchy level.

```
[edit security certificates certificate-authority]
security{
  certificates{
    certificate-authority SAE Cert{
      File /var/db/certs/cert.pem
    }
  }
}
```

2. Include the following statements at the [system services service-deployment] hierarchy level.

```
system{
  services{
    service-Deployment{
      servers {
        server-address port port-number{
          Security-options {
            tls;
          }
        }
      }
    }
  }
}
```

Creating a Client Certificate for the Router

For information about how to obtain a certificate for the router from a certificate authority, see *Obtaining a Certificate from a Certificate Authority* in the *JUNOS System Basics Configuration Guide*.

Installing the Client Certificate on the Router

To install the client (router) certificate on the JUNOS routing platform:

1. Include the following statements at the [edit security certificates certificate-authority] hierarchy level.

```
[edit security certificates certificate-authority]
security{
  certificates{
    local clientCERT { .... } ;
  }
}
```

2. Include the following statements at the [system services service-deployment] hierarchy level.

```
system{
  services{
    service-Deployment{
      local-certificate clientCert;
    }
  }
}
```

Installing the Client Certificate on the SAE

To install the client certificate on the SAE, you must import the client (router) certificate to the SAE keystore using the **keytool** command. For example:

```
/opt/UMC/jre/bin/keytool -import -alias router -file client.crt -keypass router
-noprompt -trustcacerts -keystore keystore/keystore.jks -storepass router
```

Configuring the SAE to Use TLS

To configure the SAE to accept TLS connections, enter a port number in the TLS BEEP Server Port field in the JUNOS router driver configuration.

See *Configuring the SAE to Manage JUNOS Routing Platforms* on page 144.

Configuring the Keystore for TLS Certificates and Keys

A keystore is a database of keys and certificates from trusted entities. To use SDX Configuration Editor to configure the TLS keystore on the SAE:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, expand the **JUNOS Router Driver** section, and then expand the **Keystore** section.

Keystore	
Keystore Location	keystore\keystore.jks
Keystore Password	***** Show
Need Client Authentication	Yes Disable
Keystore Implementation	JKS Disable
Certificate Algorithm	SunX509 Disable

3. Edit or accept the default values in the fields.

See *Keystore Fields for the JUNOS Router Driver* on page 152.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Keystore Fields for the JUNOS Router Driver

In SDX Configuration Editor, you can edit the Keystore fields in the JUNOS Router Driver section in the Router pane in an SAE configuration file.

Keystore Location

- Location of the keystore that contains the key/certificate pair that the SAE sends to the router. If the SAE requires client authentication, it also specifies the location of the CA certificate that was used to sign the certificate that the router sends to the SAE.
- Value—Path and name of the keystore
- Guidelines—The value of this field must match the value of the **-keystore** argument that you entered with the **keytool** command when you created the server certificate for the SAE.
See *Creating a Server Certificate for the SAE* on page 148.
- Default—keystore\keystore.jks
- Property name—Router.junos.keystore.location

Keystore Password

- Password required for the keystore.
- Value—Password; must be at least six characters
- Guidelines—The value of this field must match the value of the **-keypass** and **-storepass** arguments that you entered with the **keytool** command when you created the server certificate for the SAE.
See *Creating a Server Certificate for the SAE* on page 148.
- Default—No value
- Example—{BASE64}c2FIS2V5c3RvcmlU =
- Property name—Router.junos.keystore.storePass

Need Client Authentication

- Specifies whether or not the SAE requests a client certificate from the router.
- Value
 - Yes—The SAE asks the router for a client certificate when a connection to the router is established.
 - No—The SAE does not ask the router for a client certificate when a connection to the router is established.
- Default—Yes
- Property name—Router.junos.keystore.needClientAuth

Keystore Implementation

- Implementation type of the keystore.
- Value
 - JKS (JKS is the standard Java keystore implementation)
 - PKCS12 (Public Key Cryptography Standard #12)
- Default—JKS
- Property name—Router.junos.keystore.type

Certificate Algorithm

- Implementation type of the certificates contained in the keystore.
- Value—SUN509
- Default—SUNX509, which is the type defined in X.509 the ITU-T standard for Public Key Infrastructure (PKI).
- Property name—Router.junos.keystore.certType

Checking Changes to the JUNOS Configuration

The SAE can check the configuration of a JUNOS routing platform under its control to detect whether the configuration has changed by a means other than through the SAE. If the SAE finds a disparity between the router and the SAE configurations, it can take several actions. The SAE checks the configuration installed on the router against the state of the SAE session layer (subscriber, service, and interface sessions). While the check is occurring, the SAE does not handle jobs from the router, and all provisioning activity is blocked, including event notifications.

The SAE can take the following actions if it finds a disparity between the router and SAE configurations:

- Remove the disparate sessions from the router. When the SAE removes a session, it generates Stop events for the session and removes the session from the session store and the SAE.
- Re-create the sessions that have been removed. Subscribers whose sessions have been removed need to log back in before they can activate services. During session re-creation, the SAE responds to event notifications and provisioning operations.

If the state of the router configuration is lost because of a failover or a restart, it is not possible to re-create the sessions.

- Report disparities to the operator without making any changes to the router configuration.

The disparities are reported through the SAE router driver event trap called routerConfOutOfSynch and through the info log.

Note that it is not possible to check the consistency of individual provisioning objects. Therefore, modifications to a provisioning object while the SAE is disconnected from the router cannot be detected.

Setting Up Periodic Configuration Checking

To use SDX Configuration Editor to configure the SAE to periodically check the configuration of the JUNOS routing platform:

1. In the navigation pane, select a configuration file for the SAE that you want to configure.
2. Select the **Router** tab, expand the **JUNOS Router Driver** section, and then expand the **Configuration Checking** section.

The screenshot shows a configuration interface for 'Configuration Checking'. It has two rows. The first row is 'Configuration Checking Schedule' with a text input field containing the crontab-style string '0 0 * * * *' and an 'Enable' button. The second row is 'Configuration Checking Action' with a dropdown menu currently set to 'Enforce' and a 'Disable' button.

3. Edit or accept the default values in the fields.

See *Configuration Checking Fields for the JUNOS Router Driver* on page 154.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Configuration Checking Fields for the JUNOS Router Driver

In SDX Configuration Editor, you can modify the Configuration Checking fields in the JUNOS Router Driver section of the Router pane in an SAE configuration file.

Configuration Checking Schedule

- Specifies when the SAE checks the router configuration.
- Value—The schedule format is modeled on the UNIX crontab Entry Format (see UNIX crontab man pages). It consists of seven fields separated by space or tabs. The fields specify:
 - Minute (0-59)
 - Hour (0-23)
 - Day of month (1-31, or the first three letters of the day of month)
 - Month of the year (1-12)
 - Day of the week (0-6 with 0 = Sunday, or the first three letters of the name of the day)
 - Year (4 digits indicating the year)
 - Time Zone ID: An * indicates the SAE local time zone. For custom time zones, specify the format:
 - zone = “GMT” (“+” | “-”) (hour : minute | hour minute | hour)
 - hour = digit digit

- minute = digit digit
 - digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- Guidelines
 - An asterisk (*) is interpreted as 0 for minutes and hours and as the SAE local time zone for time zone. For all other fields, it stands for “first-last.”
 - Ranges of numbers and names are allowed. Ranges are two values separated with a hyphen. The specified range is inclusive. For example, 1-5 for the hour field specifies checking at hours 1, 2, 3, 4, and 5.
 - Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: “1,2,5,9”, “0-4,8-12”.
 - Step values can be used with ranges. Following a range with “/ < number > ” specifies skips in the number’s value through the range. For example, “0-23/2” in the hours field specifies event execution every other hour. Steps are also permitted after an asterisk, so “*/2” to specifies every 2 hours.
 - When determining the next event time based on a specific time pattern, the following rules apply:
 - Seconds and milliseconds are ignored (that is, rounded up to the closest minute).
 - If you set both a day of the month and a day of the week, only the day of month is used.
- Default—No value
- Property name—Router:junos.configcheck_schedule

Configuration Checking Action

- Action that the SAE takes when it detects disparities between the configuration of the SAE and the configuration on the router.
- Value
 - Enforce—Enforces the state of the session layer on the router. The SAE removes all sessions that have disparities and creates new sessions with the same activation parameters as the original ones.
 - Synchronize—Synchronizes the state of the session layer on the router. The SAE removes all sessions that have disparities.
 - Check only—Reports disparities through the SAE router driver event trap called routerConfOutOfSynch and through the info log. The SAE does not make any changes on the router.
- Default—Enforce
- Property name—Router:junos.configcheck_action

Using SNMP to Retrieve Information from JUNOS Routing Platforms

You can use SNMP to retrieve information from the router. For example, if you create a router initialization script that uses SNMP, you need to specify the SNMP communities that are on the router.

We recommend that you specify SNMP communities for each virtual router. (See *Adding Virtual Routers Individually* on page 139.) You can also configure global default SNMP communities.

Configuring Global SNMP Communities in the SRC Software

You can configure global default SNMP communities that are used if a VR does not exist on the router or the community strings have not been configured for the VR. To use SDX Configuration Editor to configure global default SNMP communities:

1. In the navigation pane, select a directory configuration object for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **SNMP** section.



▼ **SNMP**

Read-Only Community String *****

Read-Write Community String *****

3. Edit or accept the default values in the fields.

See *Global SNMP Community Fields* on page 156.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

Global SNMP Community Fields

In SDX Configuration Editor, you can edit the following fields in the Router pane in an SAE configuration file.

Read-Only Community String

- Default SNMP community string used for read access to the router.
- Value—SNMP community string that matches a read-only community string configured on the router
- Default—Public
- Property name—Router.read-only.community.string

Read-Write Community String

- Default SNMP community string used for write access to the router.
- Value—SNMP community string that matches a read-write community string configured on the router
- Default—Private
- Property name—Router.read-write.community.string

Developing Router Initialization Scripts

When the SAE establishes a connection with a router, it can run a router initialization script to customize the setup of the connection. Router initialization scripts are run when the connection between a router and the SAE is established and again when the connection is dropped.

We provide the IorPublisher script in the `/opt/UMC/sae/lib` folder. The IorPublisher script publishes the IOR of the SAE in the directory so that a NIC can associate a router with an SAE.

Interface Object Fields

Router initialization scripts interact with the SAE through an interface object called Ssp. The SAE exports a number of fields through the interface object to the script and expects the script to provide the entry point to the SAE.

Table 9 describes the fields that the SAE exports.

Table 9: Exported Fields

Ssp Attribute	Description
Ssp.properties	System properties object (class: java.util.Properties)—The properties should be treated as read-only by the script.
Ssp.errorLog	Error logger—Use the Ssp.errorLog.println (message) to send error messages to the log.
Ssp.infoLog	Info logger—Use the Ssp.infoLog.println (message) to send informational messages to the log.
Ssp.debugLog	Debug logger—Use the Ssp.debugLog.println (message) to send debug messages to the log.

The router initialization script must set the field Ssp.routerInit to a factory function that instantiates a router initialization object:

- `<VRName>` —Name of the virtual router in which the COPS client has been configured, format: virtualRouterName@RouterName
- `<virtualIp>` —Virtual IP address of the SAE (string, dotted decimal; for example: 192.168.254.1)
- `<realIp>` —Real IP address of the SAE (string, dotted decimal; for example, 192.168.1.20)

- `<VRIp>` —IP address of the virtual router (string, dotted decimal)
- `<transportVR>` —Name of the virtual router used for routing the COPS connection, or None, if the COPS client is directly connected

The factory function must implement the following interface:

```
Ssp.routerInit(VRName,
virtualIp,
realIp,
VRIp,
transportVR)
```

The factory function returns an interface object that is used to set up and tear down a connection for a given COPS server. A common case of a factory function is the constructor of a class.

The factory function is called directly after a COPS server connection is established. In case of problems, an exception should be raised that leads to the termination of the COPS connection.

Required Methods

Instances of the interface object must implement the following methods:

- `setup()`—Is called when the COPS server connection is established and is operational. In case of problems, an exception should be raised that leads to the termination of the COPS connection.
- `shutdown()`—Is called when the COPS server connection is terminated to the virtual router. This method should not raise any exceptions in case of problems.

Example: Router Initialization Script

The following script defines a router initialization class named *SillyRouterInit*. The interface class does not implement any useful functionality; it just writes messages to the infoLog when the router connection is created or terminated.

```
class SillyRouterInit:
    def __init__(self, vrName, virtualIp, realIp, vrIp, transportVr):
        """ initialize router initialization object """
        self.vrName = vrName
        Ssp.infoLog.println("SillyRouterInit created")

    def setup(self):
        """ initialize connection to router """
        Ssp.infoLog.println("Setup connection to VR %(vrName)s" %
                             vars(self))

    def shutdown(self):
        """ shutdown connection to router """
        Ssp.infoLog.println("Shutdown connection to VR %(vrName)s" %
                             vars(self))

#
# publish interface object to Ssp core
#
Ssp.routerInit = SillyRouterInit
```

Specifying Router Initialization Scripts on the SAE

To use SDX Configuration Editor to specify router initialization scripts:

1. In the navigation pane, select a directory configuration object for the SAE that you want to configure.
2. Select the **Router** tab, and expand the **Router Scripts** section.

Router Scripts	
Extension Path	<input type="text"/>
General Script	<input type="text"/>
JUNOS Script	<input type="text"/>
JUNOSe Script	<input type="text"/>
JUNOSe Script (XDR)	<input type="text"/>

3. Edit or accept the default values in the appropriate fields.

See *JUNOS Router Script Fields* on page 159.

4. Select **File > Save**.
5. Right-click the configuration file, select **SDX System Configuration > Export to LDAP Directory**.

JUNOS Router Script Fields

In SDX Configuration Editor, you can edit the following fields in the Router Scripts section of the Router pane in an SAE configuration file.

Extension Path

- Path to router initialization scripts that are not in the default location, */opt/UMC/sae/lib*.
- Value—List of paths separated by semicolons (;)
- Default—No value
- Property name—Extension.path

General Script

- Router initialization script that can be used for all types of routers that the SRC software supports. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Default—No value
- Property name—Router.script.*

JUNOS Script

- Router initialization script for JUNOS routing platforms. The script is run when the connection between a router and the SAE is established and again when the connection is dropped.
- Value—Name of a script
- Default—iorPublisher
- Property name—Router.script.junos

Accessing the Router CLI

You can access the CLIs of Juniper Networks routers from Policy Editor and from SDX Admin through a Telnet or SSH connection. This access allows you to display and change the configuration of the router.

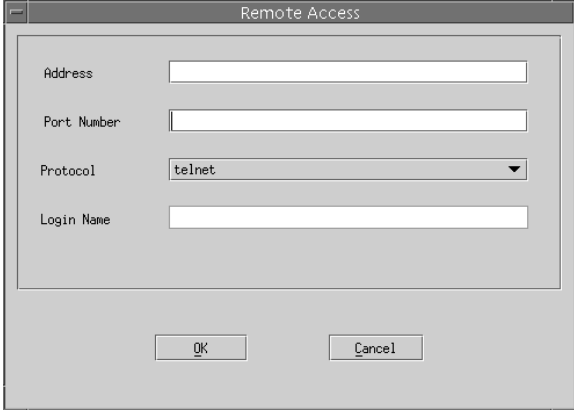
You must have the Telnet or SSH applications installed and available to Policy Editor or SDX Admin. You can open multiple Telnet or SSH sessions.

Using Policy Editor

To access a router from Policy Editor:

1. In the Policy Editor menu, select **Tools > Manage**.

The Remote Access dialog box appears.

A screenshot of the 'Remote Access' dialog box. It has a title bar with the text 'Remote Access'. Inside the dialog, there are four labeled text input fields: 'Address', 'Port Number', 'Protocol', and 'Login Name'. The 'Protocol' field is a dropdown menu with 'telnet' selected. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel'.

2. Fill in the Remote Access fields, and click **OK**.

See *Remote Access Fields* on page 161.

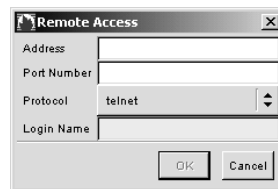
A Telnet or an SSH window with a CLI prompt appears.

Using SDX Admin

To access a router from SDX Admin:

1. In the navigation pane, expand **o = Network**.
2. Right-click on the router to which you want to connect, and select **Manage**.

The Remote Access dialog box appears.



3. Fill in the Remote Access fields, and click **OK**.

See *Remote Access Fields* on page 161.

A Telnet or an SSH window with a CLI prompt appears.

Remote Access Fields

In Policy Editor, you can edit the following fields in the Remote Access dialog box, in the select Tools > Manage menu.

In SDX Admin, you can edit the following fields in the Remote Access dialog box by right-clicking on the router object, and selecting Manage.

Address

- IP address or hostname of the router.
- Value—IP address
- Default—No value
- Example—192.0.2.1

Port Number

- TCP port over which you want to connect to the router.
- Value—TCP port
- Default—No value
- Example—22

Protocol

- Type of connection
- Value—telnet | ssh

- Default—telnet
- Example—ssh

Login Name

- Login name for SSH connections.
- Value—Text string
- Default—No value
- Guideline—You must enter a value for this property.
- Example—admin

Configuring JUNOS Routing Platforms to Interact with the SAE

To configure the JUNOS routing platform to interact with the SAE:

1. Include the following statements at the [edit system services service-deployment] hierarchy level.

```
[edit system services service-deployment]
servers server-address {
  port port-number;
}
source-address source-address;
```

2. Use the following guidelines for the variables in these statements.

server-address

- Specifies the IP address of the host on which you install the SAE.
- Value—IP address
- Guidelines—Be sure this setting matches the corresponding value in the SAE configuration.
- Default—None
- Example—192.0.2.2

port-number

- Specifies the port number for the SAE.
- Value—TCP port number
- Guidelines—Be sure this setting matches the corresponding value in the SAE configuration.
- Default—3333
- Example—3333

source-address

- Specifies the IP address of the source that sends traffic to the SAE.
- Value—IP address
- Guidelines—This setting is optional.
- Default—None
- Example—192.0.2.2

Configuring the JUNOS Routing Platform to Apply Changes It Receives from the SAE

To configure the JUNOS routing platform to receive configuration statements from the SAE and apply those statements to the configuration:

1. Create a configuration group called `sdx` that contains the configuration statements that the SAE sends to the JUNOS routing platform. To do so, include the `groups` statement at the `[edit]` level, and specify the name `sdx`.

```
[edit]
groups {
  sdx;
}
```

2. Configure the JUNOS routing platforms to apply these statements to the configuration. To do so, include the `apply-groups` statement at the `[edit]` level.

```
[edit]
set apply-groups sdx;
```

Disabling Interactions Between the SAE and JUNOS Routing Platforms

To disable the SRC software process, enter the following command.

```
root@ui1#set system processes service-deployment disable
root@ui1#commit
```

When you disable the SRC software process, it is still available on the JUNOS routing platform.

To reenable the SRC software process, enter the following command.

```
root@ui1#delete system processes service-deployment disable
root@ui1#commit
```

The SRC software process attempts to reconnect the JUNOS routing platform to the SAE.

Monitoring Interactions Between the SAE and JUNOS Routing Platforms

Use the following command on JUNOS routing platforms to monitor the connection between the JUNOS routing platform and the SAE.

```
root@ui1> show system services service-deployment
Connected to 172.17.20.151 port 3333 since 2004-02-06 14:50:31 PST
Keepalive settings: Interval 15 seconds
Keepalives sent: 100, Last sent: 6 seconds ago
Notifications sent: 0
Last update from peer: 00:00:06 ago
```

You can also monitor the interactions between the SRC software and JUNOS routing platforms in the log files for the SAE and in the log files generated by the SRC software process on the JUNOS routing platform. For information about configuring logging for the SAE, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 4, Configuring Logging for SRC Components on a Solaris Platform*. For information configuring logging on JUNOS routing platforms, see *JUNOS System Basics Configuration Guide*.

Troubleshooting SRC Problems on JUNOS Routing Platforms

To troubleshoot SRC problems on the JUNOS routing platform, review the log files for the SAE and the log files generated by the SRC software process on the router.

- If the log files indicate that the SRC software process is not responding, see *Troubleshooting Problems with the SRC Software Process* on page 164.
- If the log files indicate a problem with a specific interface, see *Troubleshooting Problems with Interfaces* on page 165.
- If the log files indicate a problem with a specific service or its associated firewall rules, see *Troubleshooting Problems with Services* on page 168.

Troubleshooting Problems with the SRC Software Process

If the log files indicate that the SRC software process is not responding:

1. Look at the status of the process on the JUNOS routing platform.

```
root@ui1> show system services service-deployment
Connected to 172.17.20.151 port 3333 since 2004-02-06 14:50:31 PST
Keepalive settings: Interval 15 seconds
Keepalives sent: 100, Last sent: 6 seconds ago
Notifications sent: 0
Last update from peer: 00:00:06 ago
```

2. If you see the message “error: the service-deployment subsystem is not running,” reenable the SRC software process (see *Disabling Interactions Between the SAE and JUNOS Routing Platforms* on page 163).
3. If the process is already enabled, review the configurations of the router and the SAE in the directory, and fix any problems.

4. Restart the SRC software process on the router.

```
root@ui1>restart service-deployment
```

The SAE synchronizes with the SRC software process and deletes unnecessary data from the router.

Troubleshooting Problems with Interfaces

If the log files indicate a problem with a specific interface or its associated firewall rules:

1. Review the configuration of the policies associated with the interfaces with the C-Web interface.
 - a. Select **SAE** from the side pane, and click **Policies**.

The Policies pane appears.

The screenshot shows the Juniper C-Web interface. On the left is a navigation pane with a tree structure. The 'SAE' node is selected, and the 'Policies' sub-pane is active. The main content area displays the 'Policies' configuration form. At the top right, it says 'Logged in as: admin' and has links for 'About', 'Refresh', and 'Logout'. Below the navigation pane, the 'Policies' section has a title bar with 'SAE' and 'Policies'. The form contains three input fields: 'Policy Group' (a text box), 'Style' (a dropdown menu), and 'Maximum Results' (a text box). To the right of these fields are three informational boxes: 'Name of a policy group. Please enter: All or part of the policy group name', 'Output style. Choices: brief: Display only policy group names', and 'Number of results to be displayed. Legal range: 1 .. INF Default value: 25'. At the bottom of the form are 'OK' and 'Reset' buttons. The footer of the interface shows 'Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy.' and the 'Juniper Your Net.' logo.

For more information on these fields, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 14, Monitoring SAE Data with the C-Web Interface*.

- b. Click **Execute**.

The Policies pane displays the interfaces available on the router.

- c. If you find any errors, fix the configuration in the directory, and proceed to Step 5. Otherwise, proceed to Step 2.

2. Review the configuration of interfaces on the JUNOS routing platform with the C-Web interface.

- a. Select **SAE** from the side pane, and click **Interfaces**.

The Information About Router Interfaces pane appears.

Monitor Logged in as: admin About Refresh Logout SAE > Interfaces

ACP
CLI
Component
Date
Disk
Interfaces...
JPS
NIC
NTP
Redirect Server
Route...
SAE
Security
System

SAE
Interfaces

Interface Name	<input type="text"/>	Name of router interface. <i>Please enter: All or part of the interface name</i>
Virtual Router	<input type="text"/>	Name of virtual router. <i>Please enter: All or part of the virtual router name</i>
Style	<input type="text"/>	Output style. <i>Choices:</i> brief: Display only interface names
Maximum Results	<input type="text"/>	Number of results to be displayed. <i>Legal range: 1 .. INF</i> <i>Default value: 25</i>

OK Reset

Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy. Juniper Your Net.

- b. Click **Execute**.

The Information About Router Interfaces pane displays the interfaces available on the router.

- c. If you find any errors, fix the configuration in the directory, and proceed to Step 5. Otherwise, proceed to Step 2.

3. Display the corresponding interfaces on the JUNOS routing platform.

```
root@olive1# show groups sdx interfaces
<fe-0/0/0> {
  unit <0> {
    family inet {
      filter {
        input SDX_PRIVATE_ID0000000000001092282;
        output SDX_PRIVATE_ID0000000000001223352;
      }
    }
  }
}
```

If you find any errors, fix the configuration in the directory, and proceed to Step 5. Otherwise, proceed to Step 4.

4. Remove the configuration for this interface from the JUNOS routing platform.

- a. Disable the SRC software process.

```
root@ui1#set system processes service-deployment disable
root@ui1#commit
```

- b. Delete the interfaces from the router.

```
delete groups sdx interfaces <interfaceName> <interfaceIdentifier>
root@ui1#commit
```

For example, to delete the interface with identifier fe-0/0/0 unit 0, enter:

```
root@ui1#delete groups sdx interfaces <fe-0/0/0> unit <0>
root@ui1#commit
```

- c. Reenable the SRC software process.

```
root@ui1#delete system processes service-deployment disable
root@ui1#commit
```

5. Restart the SRC software process on the router.

```
root@ui1>restart service-deployment
```

The SAE reconfigures the interface that you deleted.

6. Review the log files again.

If the action you took did not fix the problem, return to the last step you performed, and proceed with this troubleshooting procedure. If you have performed all the tasks in the troubleshooting procedure and the problem persists, delete all SRC data on the JUNOS routing platform (see *Deleting All SRC Data on JUNOS Routing Platforms* on page 171).

Troubleshooting Problems with Services

If the log files indicate a problem with a specific service or its associated firewall rules:

1. Review the configuration of the policies associated with the interfaces with C-Web.
 - a. Select **SAE** from the side pane, and click **Policies**.

The Policies pane appears.

The screenshot shows the Juniper C-Web interface. On the left is a sidebar menu with items: Monitor, ACP, CLI, Component, Date, Disk, Interfaces..., JPS, NIC, NTP, Redirect Server, Route..., SAE, Security, and System. The 'SAE' item is selected. The main content area is titled 'SAE' and 'Policies'. It contains three input fields: 'Policy Group' (a text box), 'Style' (a dropdown menu), and 'Maximum Results' (a text box). To the right of these fields are help text boxes: 'Name of a policy group. Please enter: All or part of the policy group name', 'Output style. Choices: brief: Display only policy group names', and 'Number of results to be displayed. Legal range: 1 .. INF Default value: 25'. Below the input fields are 'OK' and 'Reset' buttons. At the top right of the main area, it says 'Logged in as: admin' and has links for 'About', 'Refresh', and 'Logout'. At the bottom left, it says 'Copyright © 2007, Juniper Networks, Inc. All Rights Reserved. Trademark Notice. Privacy.' and at the bottom right is the 'Juniper Your Net.' logo.

For more information on these fields, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 14, Monitoring SAE Data with the C-Web Interface*.

- b. Click **Execute**.

The Policies pane displays the interfaces available on the router.

- c. If you find any errors, fix the configuration in the directory, and proceed to Step 5. Otherwise, proceed to Step 2.
2. Review the configuration of the service on the JUNOS routing platform with C-Web.
 - a. Select **SAE** from the side pane, and click **Services**.

The State of Services Running on the SAE pane appears.

Monitor		Logged in as: admin		About	Refresh	Logout
ACP	SAE	SAE > Services				
CLI	Services					
Component	Service Name	<input type="text"/>	Name of service. <i>Please enter: All or part of the service name</i>			
Date	Secret	<input type="checkbox"/>	Display subscriber sessions and service sessions for hidden services.			
Disk	Style	<input type="text"/>	Output style <i>Choices:</i> brief: Display only service names			
Interfaces...	Maximum Results	<input type="text"/>	Number of results to be displayed. <i>Legal range: 1 .. INF</i> <i>Default value: 25</i>			
JPS	OK Reset					
NIC						
NTP						
Redirect Server						
Route...						
SAE						
Security						
System						

Copyright © 2007, Juniper Networks, Inc. [All Rights Reserved](#). [Trademark Notice](#). [Privacy](#). Juniper your Net.

For more information on these fields, see *SRC-PE Monitoring and Troubleshooting Guide, Chapter 14, Monitoring SAE Data with the C-Web Interface*.

- b. Click **Execute**.

The State of Services Running on the SAE pane displays the interfaces available on the router.

- c. Locate an active service session for this service, and observe the ProvisioningSet field of that session.
- d. Locate an identifier that is associated with the service that is causing the problem.

For example, in the above display, the identifier SDX_PRIVATE_ID0000000000002075317 is associated with a Network Address Translation (NAT) rule.

3. Review the corresponding configuration on the JUNOS routing platform.

```

root@olive1# show groups sdx services nat rule SDX_PRIVATE_ID00000000
000002075317
    match-direction input;
    term SDX_PRIVATE_TERM {
        from {
            source-address {
                0.0.0.0/0;
            }
            destination-address {
                0.0.0.0/0;
            }
        }
        then {
            translated {
                source-pool SDX_PRIVATE_ID00000000000002009780;
                translation-type source dynamic;
            }
        }
    }
}

```

If you find any errors, fix the configuration in the directory and proceed to Step 5. Otherwise, proceed to Step 4.

4. Remove the configuration for this service from the JUNOS routing platform.
 - a. Disable the SRC software process.

```

root@ui1#set system processes service-deployment disable
root@ui1#commit

```

- b. Delete the service on the JUNOS routing platform.

```

delete groups sdx services <serviceName> <filterID>
root@ui1#commit

```

For example, to delete a firewall filter of the service called firewall with filterID SDX_PRIVATE_ID0000000000001223352, enter:

```

delete groups sdx services firewall filter
SDX_PRIVATE_ID0000000000001223352
root@ui1#commit

```

- c. Reenable the SRC software process.

```

root@ui1#delete system processes service-deployment disable
root@ui1#commit

```


- Restart the SRC software process on the JUNOS routing platform.

```
root@ui1>restart service-deployment
```

The SAE reconfigures the service that you deleted on the JUNOS routing platform.

- Review the log files again.

If the action you took did not fix the problem, return to the last step you performed, and proceed with this troubleshooting procedure. If you have performed all the tasks in the troubleshooting procedure and the problem persists, delete all SRC data on the JUNOS routing platform (see *Deleting All SRC Data on JUNOS Routing Platforms* on page 171).

Deleting All SRC Data on JUNOS Routing Platforms

If deleting parts of the SRC data on a JUNOS routing platform fails to solve problems, delete all the SRC data and restart the SRC software process. To do so:

- Delete all SRC interfaces and services.

```
delete groups sdx  
root@ui1#commit
```

- If you are running SDX software releases 5.0 through 6.1, you should also delete interface sessions. (After release 6.2, session data is no longer stored on the router, it is stored on the SAE host using the session store feature.)

```
delete groups sdx-sessions  
root@ui1#commit
```

- Restart the SRC software process on the router.

```
root@ui1>restart service-deployment
```

The SAE reconfigures all the interfaces and services that you deleted from the router.

Part 3

**Locating Subscriber Management
Information**

Chapter 9

Locating Subscriber Information with the NIC

This chapter describes the network information collector (NIC) that the SRC software uses to locate subscriber information for an application and discusses strategies for implementing a NIC configuration. The chapter includes information about the NIC sample data provided with the SRC software; reviewing this data will help you plan a NIC configuration for your network. Topics include:

- Locating Subscriber Management Information on page 175
- Mapping Subscribers to a Managing SAE on page 177
- High Availability for NIC on page 178
- Planning a NIC Implementation on page 181
- NIC Agents Used in the NIC Configuration Scenarios on page 184
- Router Initialization Scripts with NIC Configuration Scenarios on page 186

Locating Subscriber Management Information

For services to be activated for a subscriber session, applications such as the SRC Volume-Tracking Application (SRC-VTA), Dynamic Service Activator, Enterprise Manager Portal, or a residential portal need to locate the SAE that manages the subscriber. An application such as the Threat Mitigation Application Portal needs to locate the SAE that manages interfaces through which traffic destined for a specified IP address enters the network.

The NIC is the component that locates which SAE manages a subscriber or an interface. The NIC uses information that identifies the subscriber or the interface to identify the managing SAE. A NIC is similar to a Domain Name System (DNS) in that a NIC processes resolution requests. Rather than translating hostnames to IP addresses and vice versa, the NIC resolves an identifier for a subscriber or an interface to a reference for the managing SAE.

The components that participate in this resolution are a NIC host and a NIC proxy, also called a NIC locator for particular applications. A NIC host processes resolution requests. A NIC proxy requests data resolution for an application. A NIC proxy is so-named because it requests information on behalf of an application. A NIC proxy and a NIC host communicate with each other through Common Object Request Broker Architecture (CORBA); NIC manages the CORBA interactions for you.

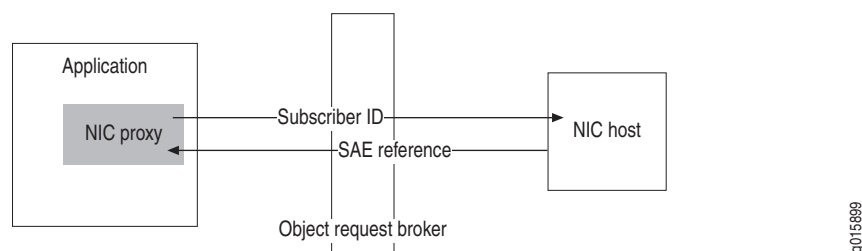
NIC can operate in a client/server mode or in a local host mode. In the client/server mode, a NIC host and NIC proxies can reside on different systems. In local host mode, a NIC host and NIC proxies reside in the same process on a machine.

NIC Client/Server Mode

In client/server mode, a NIC host is the server. A NIC proxy, which comprises libraries within an application that interacts with a NIC host, is the client.

Figure 7 shows a NIC proxy running within an application and a NIC host running on a different machine. Both communicate through CORBA, with the NIC proxy providing an identifier for a subscriber and the NIC host returning a reference to the SAE that manages the subscriber.

Figure 7: Communication Between a NIC Proxy and a NIC Host in Client/Server Mode



NIC Local Host Mode

In local host mode, a Java application can include the libraries for a NIC host as well as NIC proxies. With this configuration, the NIC host and the NIC proxies communicate with each other within the same application. Because both components run within the same application, the application and the NIC host start and stop at the same time.

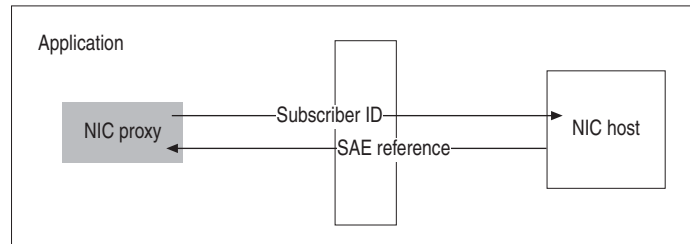
If an application uses a local NIC host, all NIC proxies for the application typically communicate with the local NIC host, but some of the NIC proxies can be configured to communicate with a NIC host that runs on another system.

When you use NIC in local host mode:

- You cannot use C-Web to monitor or troubleshoot the local NIC host
- The NIC host runs all the resolvers and agents for the host on the local machine.
- Other NIC hosts cannot communicate with agents and resolvers that run in a local NIC host.

Figure 8 shows a NIC proxy and a NIC host running within an application.

Figure 8: Communication Between a NIC Host and a NIC Proxy in Local Host Mode



g015957

Mapping Subscribers to a Managing SAE

A NIC collects information about the state of the network and can provide mapping from a specified type of network data, known as a *key*, to another type of network data, known as a *value*. Applications can use a NIC proxy to submit a key to a NIC host. The NIC host obtains a corresponding value from other components within NIC and returns it through the NIC proxy to the application. A typical use of a NIC is for a residential portal application to submit a subscriber's IP address and for the NIC to return the interoperable object reference (IOR) of the SAE managing that subscriber.

NIC Proxies and NIC Locators

Typically, an application supports one NIC proxy for each type of data request. A NIC proxy caches resolution results for a period of time so that it can resolve future requests without consulting the NIC host, thereby decreasing traffic between the NIC proxy and the NIC host. Applications that use NIC proxies communicate with the proxy to delete any invalid cache entries. Caching lets you optimize resolution performance for your network configuration and system resources.

You configure a NIC proxy when you configure that application. SRC applications such as the SRC-VTA and Dynamic Service Activator contain NIC proxies. If you are writing an external application that will interact with a NIC, you must include NIC proxies in the application.

A NIC locator provides the same functionality as a NIC proxy; however, it runs as part of the NIC host. A NIC locator uses the NIC access interface module, a simple CORBA interface, to enable non-Java applications to interact with NIC. A NIC locator does not cache information.

For information about the NIC access interface module, see the API documentation in the SRC software distribution in the folder *SDK/doc/idl/nic* or on the Juniper Networks Web site at

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

For more information about NIC proxies and NIC locators, see *Chapter 16, Developing Applications That Use NIC*.

NIC Hosts

NIC hosts collect and store SRC information, and respond to requests from NIC proxies. The components in a NIC host that manage this process are:

- NIC agents—Collect data from SRC components, publish data, and make data available to NIC resolvers
- NIC resolvers—Process resolution requests

NIC Agents

NIC agents collect information about the state of the network from many data sources on the network. Table 10 describes the types of agents supplied with NIC.

Table 10: Types of NIC Agents

Type of Agent	Type of Information the Agent Makes Available
Consolidator agent	Summary information received from other agents.
Directory agent	Specified directory entries and changes to directory entries.
Properties agent	Information from a specified list of property file. Typically, you do not configure properties agents.
SAE plug-in agent	Subscriber information and interface information for SAE-managed subscribers and interfaces.
XML agent	Information from a specified XML document. Typically, you do not configure XML agents.

NIC Resolvers

NIC resolvers manage information to resolve requests by:

- Receiving and storing information about the state of the network from components within NIC and other NIC resolvers
- Requesting information from NIC agents and other NIC resolvers
- Receiving requests from the NIC proxies or other NIC resolvers
- Processing requests and sending responses to the requesters

High Availability for NIC

You can configure high availability for NIC when you use client/server mode with the NIC host and the NIC proxies running on different machines. NIC supports several mechanisms to maintain high availability. We recommend that you use NIC replication to keep a NIC configuration highly available. NIC replication uses groups of NIC hosts that share the same configuration for NIC resolutions to respond to resolution requests.

When you use NIC in local host mode, you do not need to configure redundancy for a NIC host, because the NIC host runs within the application.

High Availability in Existing NIC Configurations

If you have a previous NIC configuration, you may be using:

- NIC host redundancy, in which a set of NIC hosts provide redundancy

The SRC CLI does not support NIC host redundancy.

- Redundancy for SAE plug-in agents, in which a set of SAE plug-in agents provide redundancy

If you have an SAE plug-in agent that uses agent redundancy, enable state synchronization for the agent and use NIC replication. In SRC Release 1.0.0, configuration for SAE plug-in agent redundancy is discontinued.

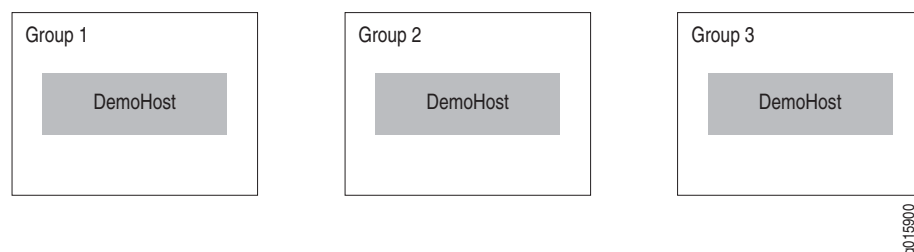
NIC Replication

NIC replication uses the concept of a group to identify a NIC host that has a particular configuration. A group contains one or more NIC hosts; each NIC host in a group is unique; for example, each NIC host could reside on a different system. A NIC proxy contacts specified groups that contain hosts with the same configuration to locate a managing SAE.

For example, a group might include the host DemoHost, but not two instances of DemoHost. Typically, each NIC host in a group is located in the same point of presence (POP). However, a machine can support only one NIC host. The SRC software stores groups in the directory in *ou = dynamicConfiguration*, *ou = Configuration*, *o = Management*, *o = umc*.

For example, Figure 9 shows three NIC groups with each group containing a NIC host that has the same configuration.

Figure 9: NIC Groups



Groups let you:

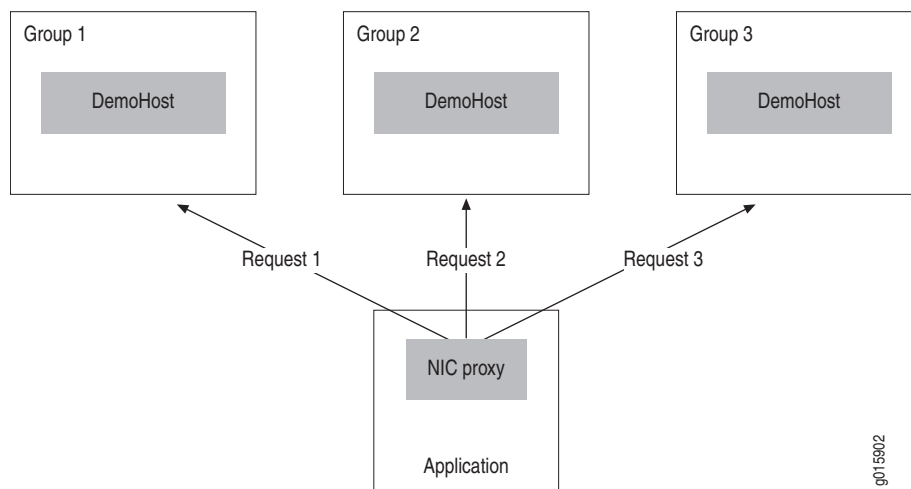
- Distribute network and processing load between two or more groups
- Provide failover protection if one group becomes unavailable

With NIC replication, a NIC proxy can contact multiple NIC hosts that are assigned to different groups. When a NIC proxy is configured to contact more than one group, the NIC configuration on a NIC host in each group should be equivalent—the NIC hosts should use the same configuration scenarios.

A NIC proxy selects a group by using the method specified in the configuration for the proxy; for example, the NIC proxy can randomly choose a group from a list. The NIC proxy then sends resolution requests to the corresponding host in that group. If a NIC proxy submits high numbers of resolution requests to the NIC host, you can configure the NIC proxy to randomly pick a NIC host or to pick a NIC host in a cyclic order to decrease the probability that one NIC host manages all the resolution requests.

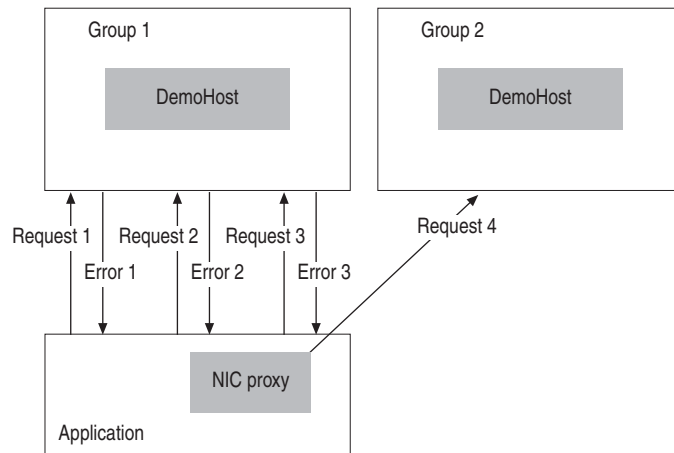
Figure 10 shows resolution requests sent by means of a round-robin selection.

Figure 10: NIC Group Selection by Round-Robin



If the NIC host fails to respond to a specified number of resolution requests, the NIC proxy stops sending resolution requests to the unavailable NIC host and sends the resolution requests to another NIC host. The NIC proxy continues to poll the unavailable NIC host to determine its availability. When the NIC host becomes available, the NIC proxy can again send resolution requests to that host.

Figure 11 shows a NIC proxy that sends a resolution request to Group 1, receives an error message, then sends two more resolution requests before sending a request to Group 2 rather than Group 1. When Group 1 is available again, the NIC proxy will send the request to Group 1.

Figure 11: NIC Resolution Request

g015903

You configure NIC replication for hosts, then configure NIC proxies to use replication.

Although you can distribute agents and resolvers among different hosts, as shown in the configuration for the NIC hosts OnePopBO and OnePopH1 in the sample data, we recommend that you use the DemoHost configuration, which centralizes the configuration for agents and resolvers.

Planning a NIC Implementation

The SRC software provides standard NIC configuration scenarios that you can modify to meet the requirements for your environment. Which scenarios you choose depends on the applications you use.

If the resolution scenarios do not provide the type of resolution needed, we recommend that you consult Juniper Professional Services.

If you want to customize configuration of the scenarios provided for a NIC running on a Solaris platform, see *Chapter 19, Customizing a NIC Configuration*.

To plan your NIC implementation:

1. Review the NIC configuration scenarios, and select the scenario that best fits the requirements for your application. In most cases, one of the basic configuration scenarios provides the type of resolution needed.

See *NIC Configuration Scenarios* on page 182.
2. Determine the number of NIC proxies that you will need to access NIC hosts, and estimate the amount of traffic between the NIC proxies and the NIC hosts. If you expect heavy traffic between NIC proxies and NIC hosts, configure a number of NIC hosts to share the traffic load and processing.

3. Determine which NIC hosts to assign to a group to provide NIC replication; choose names for these groups.
4. If you have not done so already, determine which systems are to run NIC hosts.

NIC Configuration Scenarios

Table 11 lists the NIC configuration scenarios provided in the SRC software.

Table 11: NIC Configuration Scenarios

Configuration Scenario	Name of NIC Configuration Scenario to Use	Type of Resolution	Notes
Basic Configuration Scenarios			
For JUNOS local configuration for PPP and DHCP subscribers. Sample use: DSL providers for residential customers.	OnePop	Subscriber IP address to the SAE IOR	Simplest configuration. IP pools configured locally on each virtual router (VR) with IP addresses from a static pool of IP addresses configured on the virtual router.
For subscribers who have an accounting ID. Can be used for multiple subscribers who use the same accounting ID, in which case NIC returns all SAE IORs for mapped subscribers. Sample use: Support for the volume-tracking application.	OnePopAcctId	Accounting ID of a subscriber to the SAE IOR and the IP address of a subscriber to accounting ID	A subscriber's accounting ID can be specified at subscriber login from the SAE subscriber classification script. As a result, the accounting ID encapsulates other attributes of the subscriber session processed by the subscriber classification script. The OnePopAcctId configuration scenario can resolve the encapsulated attributes. For example, customers can assign a subscriber username (login id without domain name) to an accounting ID with the following subscriber classification. [< -retailerDn- > ?accountingU serId = < -userName- > ?sub?(uniqueID = < -userName- >)]
For subscribers who have assigned IP addresses (assigned external to the SAE). Sample use: In a PacketCable Multimedia Specification (PCMM) environment when the SAE acts as both a policy server and application manager.	OnePopDynamicIp	Subscriber IP address to the SAE IOR	

Table 11: NIC Configuration Scenarios (continued)

Configuration Scenario	Name of NIC Configuration Scenario to Use	Type of Resolution	Notes
<p>For resolution of a subscriber login name to an SAE IOR, and of a subscriber IP address to a subscriber login name.</p> <p>Sample use:</p> <p>Support for tracking subscriber bandwidth usage or for using a billing model. You can use the SRC-VTA with this scenario.</p>	OnePopLogin	Subscriber login name to the SAE IOR and subscriber IP address to login name	Uses two resolvers. Use a separate NIC proxy for each resolution.
<p>For subscribers who connect through a cable modem termination system (CMTS) device.</p> <p>Sample use:</p> <p>In a PCMM environment in which the policy server is separate from the application server. This scenario can be used when the configuration includes Juniper Policy Server or another policy server, and the SAE is an application manager.</p>	OnePopPcmm	Subscriber IP address to the SAE IOR	
<p>For use with applications that use the SAE programming interfaces and that identify subscribers by the primary username.</p> <p>Sample uses:</p> <ul style="list-style-type: none"> ■ Aggregate services ■ Dynamic service activator application 	OnePopPrimaryUser	Primary username of a subscriber to the SAE IOR	Similar to <i>OnePopLogin.xml</i> .
<p>For a router configuration in which VRs share IP pools.</p> <p>Sample use:</p> <ul style="list-style-type: none"> ■ Services for enterprise subscribers. ■ Support for two different proxies: <ul style="list-style-type: none"> ■ Subscriber DN to the SAE IOR ■ Subscriber IP address to the SAE IOR 	OnePopDnSharedIp	Subscriber distinguished name (DN) or subscriber IP address to the SAE IOR	Includes resolution available in <i>OnPopSharedIp.xml</i> and adds resolution from a subscriber DN.

Table 11: NIC Configuration Scenarios (continued)

Configuration Scenario	Name of NIC Configuration Scenario to Use	Type of Resolution	Notes
For a router configuration in which pools can be shared among routers. Pools can be assigned by RADIUS or by a DHCP server. Sample use: Support for DHCP and PPP connections for residential subscribers.	OnePopSharedIp	Subscriber IP address to the SAE IOR	
For scenarios in which subscribers have an assigned IP address and these IP addresses can be associated with interfaces on JUNOS routing platforms. Sample use: ■ Threat Mitigation Application Portal	OnePopStaticRouteIp	Assigned subscriber IP address to the SAE IOR	Static route information for routers resides in an XML document in the directory under the router object.
For enterprise customers.	OnePopAllRealms	Subscriber IP address or subscriber DN to the SAE IOR	The scenario combines the OnePop and OnePopSharedIp scenarios and adds resolution from a subscriber DN.
Advanced Configuration Scenario			
For two POPs that share a back office. Sample use: Support for a deployment that has a back office that connects to NIC hosts at other sites.	MultiPop	Subscriber IP address to the SAE IOR	You can deploy this scenario in an environment that has a number of POPs; for example, a configuration in which there are two POPs with NIC proxy communication to a back office, which in turn communicates with the POP hosts. The POP hosts each support parallel hosts and agents and manage resolutions in the same way. You can add POPs by copying the configuration for one POP and modifying the configuration to suit your environment.

NIC Agents Used in the NIC Configuration Scenarios

When you configure a NIC configuration scenario, you use the basic configuration for each NIC agent in the scenario, but modify properties such as directory properties to make the agent configuration compatible with your SRC configuration. The NIC configuration scenario that you use determines which agents appear in your configuration.

Table 12 lists all agents that are available in the various configuration scenarios.

Table 12: NIC Agents

Agent Name	Type of Agent	Type of Information
AcctIdIp	SAE plug-in	Mappings of accounting IDs of a subscribers to the SAE IOR and subscriber IP addresses to accounting ID(s)
DnVr	SAE plug-in	Mappings of enterprise access DNs to VRs
Enterprise	Directory	List of enterprise names
IpAcctId	SAE plug-in	Mappings of subscriber IP addresses to accounting IDs
IpLoginName	SAE plug-in	Mappings of IP addresses to login names
IpVr	SAE plug-in	Mappings of IP addresses to VRs
LoginNameVr	SAE plug-in	Mappings of login names to VRs
PoolVr	Directory	Mappings of IP pools to VRs
UserNameVr	SAE plug-in	Mappings of subscriber IP addresses to accounting IDs
VrSaeId	Directory	Reads information about virtual routers and the mappings between virtual routers and SAEs

Table 13 shows the types of agents that each configuration scenario uses.

Table 13: Agents in Configuration Scenarios

NIC Configuration Scenario	Directory Agents	SAE Plug-In Agents
OnePop	PoolVr, VrSaeId	
OnePopAcctId	PoolVr, VrSaeId	AcctIdIp, IpAcctId
OnePopDnSharedIp	PoolVr, VrSaeId, Enterprise	DnVr
OnePopDynamicIp	PoolVr, VrSaeId	
OnePopLogin	Pool, VrSaeId	IpLoginName, LoginNameVr
OnePopPcmm	PoolVr, VrSaeId	
OnePopSharedIp	PoolVr, VrSaeId	IpVr
MultiPop	PoolVr, VrSaeId, site-specific versions of PoolVr and VrSaeId	IpVr
OnePopAllRealms	PoolVr, VrSaeId, Enterprise	IpVr
OnePopPrimaryUser	VrSaeId	UserNameVr
OnePopStaticRouteIp	VrSaeId, PoolInterface	



NOTE: If you use a configuration scenario that includes an SAE plug-in agent, make sure that your network has a CORBA naming server that includes the names of the servers that host the SAE plug-in agents. The SRC software distribution includes a CORBA naming server in the omniORB package.

Router Initialization Scripts with NIC Configuration Scenarios

The NIC resolutions map VRs to SAEs. For these resolutions, use a router initialization script that associates each VR with the SAE that manages it. Which router initialization script you use depends on whether the SAE obtains IP pools from JUNOS VRs:

- **poolPublisher** router initialization script—Use when the SAE obtains local IP pools locally from JUNOS VRs.
- **iorPublisher** router initialization script—Use when the router is one of the following:
 - JUNOS routers that do not supply IP addresses from local pools
 - JUNOS routing platforms
 - CMTS devices

These devices do not supply IP addresses from local pools in your network.

Table 14 lists which type of initialization script should be used with the various NIC configuration scenarios.

Table 14: Type of Router Initialization Script to Use for NIC Configuration Scenarios

poolPublisher	iorPublisher	poolPublisher or iorPublisher
One Pop	OnePopDnSharedIp	OnePopAcctId
	OnePopPcmm	OnePopAllReams
	OnePopPrimaryUser	OnePopDynamicIp
	OnePopSharedIp	OnePopLogin
	OnePopStaticRouteIp	MultiPop



NOTE: If you modify information about IP pools on a VR after the COPS connection is established, the SAE does not automatically register the changes, and you must update the directory.

For more information about router initialization scripts for JUNOS routers, including how to update the directory, see *Chapter 7, Using JUNOS Routing Platforms in the SRC Network with the SRC CLI*.

For more information about router initialization scripts for JUNOS routing platforms, see *Chapter 7, Using JUNOS Routing Platforms in the SRC Network with the SRC CLI*.

Chapter 10

Configuring NIC with the SRC CLI

This chapter describes how you can use the SRC CLI to configure the network information collector (NIC). You can use the CLI to configure the NIC on a Solaris platform or on a C-series platform.

You can also use SRC configuration applications to configure the NIC on a Solaris platform. See *Chapter 11, Configuring NIC on a Solaris Platform*.

Topics in this chapter include:

- Configuration Statements for the NIC on page 188
- Before You Configure the NIC on page 190
- Configuring the NIC from the SRC CLI on page 191
- Starting the NIC from the SRC CLI on page 191
- Reviewing and Changing Operating Properties for NIC on page 192
- Configuring NIC Replication on page 194
- Configuring a NIC Scenario on page 194
- Verifying Configuration for the NIC on page 203
- Testing a NIC Resolution on page 203
- Stopping a NIC Host on a C-series Platform on page 203
- Restarting the NIC on page 204
- Restarting a NIC Agent on page 204
- Restarting a NIC Resolver on page 204
- Changing NIC Configurations on page 204

For overview information about the NIC, see *Chapter 9, Locating Subscriber Information with the NIC*.

Configuration Statements for the NIC

The SRC CLI provides the following groups of configuration statements for the NIC:

- *Configuration Statements for NIC Operating Properties* on page 188
- *Configuration Statements for NIC Scenarios* on page 189
- *Configuration Statements for NIC Logging* on page 190



NOTE: We recommend that you change only those statements visible at the basic editing level. Contact Juniper Professional Services or Juniper Customer Support before you change any of the NIC statements and options not visible at the basic editing level.

Configuration Statements for NIC Operating Properties

Use the following configuration statements to configure the NIC operating properties at the [edit] hierarchy level. These statements are visible at the CLI basic editing level.

```
slot number nic {
    base-dn base-dn;
    java-heap-size java-heap-size;
    snmp-client;
    hostname hostname;
    runtime-group runtime-group;
}

slot number nic initial {
    static-dn static-dn;
    dynamic-dn dynamic-dn;
}

slot number nic initial directory-connection {
    url url;
    backup-urls [backup-urls...];
    principal principal;
    credentials credentials;
    protocol (ldaps);
    timeout timeout;
    check-interval check-interval;
    blacklist;
    snmp-agent;
}
```

```

slot number nic initial directory-eventing {
    eventing;
    signature-dn signature-dn;
    polling-interval polling-interval;
    event-base-dn event-base-dn;
    dispatcher-pool-size dispatcher-pool-size;
}

```

For detailed information about each configuration statement, see the *SRC-PE CLI Command Reference*.

Configuration Statements for NIC Scenarios

Use the following configuration statements to configure the NIC at the [edit] hierarchy level. These statements are visible at the CLI basic editing level.

Which agents you configure depends on the NIC configuration scenario that you use.



NOTE: The CLI also provides configuration statements for consolidator agents, properties agents, and XML agents. At this time, none of the NIC configuration scenarios uses these agents. The following list does not include the configuration statements for these agents.

```
shared nic scenario name
```

```
shared nic scenario name agents name
```

```

shared nic scenario name agents name configuration directory {
    search-base search-base;
    search-filter search-filter;
    search-scope (0 | 1 | 2);
    server-url server-url;
    directory-backup-urls directory-backup-urls;
    principal principal;
    credentials credentials;
}

```

```

shared nic scenario name agents agent configuration sae-plugin {
    event-filter event-filter;
    number-of-events number-of-events;
}

```

For detailed information about each configuration statement, see the *SRC-PE CLI Command Reference*.

Configuration Statements for NIC Logging

Use the following configuration statements to configure logging for the NIC at the [edit] hierarchy level.

```
shared nic scenario name hosts name configuration logger name syslog {
  filter filter;
  host host;
  facility facility;
  format format;
}
```

```
shared nic scenario name hosts name configuration logger name file {
  filter filter;
  filename filename;
  rollover-filename rollover-filename;
  maximum-file-size maximum-file-size;
}
```

Before You Configure the NIC

When you use NIC in a client/server configuration, you configure the NIC scenario before you configure the NIC proxies.

Before you configure NIC hosts from the CLI:

- Plan your NIC implementation:
 - Choose the NIC configuration scenario to use.

The default scenario is OnePop.

For information about NIC configuration scenarios and NIC agents, see *Chapter 9, Locating Subscriber Information with the NIC*.

- If you are using the CLI on a Solaris platform, install the NIC data. If you are using the CLI on a C-series platform, the NIC data is already installed.

For information about installing the NIC sample data on a Solaris platform, see *SRC-PE Getting Started Guide, Chapter 29, Defining an Initial Configuration on a Solaris Platform*.

- Ensure that the appropriate type of router initialization script is configured for the router or network device.

See *Chapter 9, Locating Subscriber Information with the NIC*.

- Set the editing level for the CLI to basic. This ensures that only the statements that you need to configure are visible.

See *SRC-PE CLI User Guide, Chapter 9, Controlling the CLI Environment*.

Configuring the NIC from the SRC CLI

Before you configure the NIC, complete the prerequisite tasks.

See *Before You Configure the NIC* on page 190.

To configure the NIC:

1. Start the NIC component.

See *Starting the NIC from the SRC CLI* on page 191.

2. Configure NIC operating properties.

See *Reviewing and Changing Operating Properties for NIC* on page 192.

3. Configure NIC replication.

See *Configuring NIC Replication* on page 194.

4. (Optional) If you plan to use a configuration scenario other than OnePop (the default), delete any data for the OnePop scenario and configure the local static DN to specify the configuration scenario.

See *Changing NIC Configurations* on page 204.

5. Configure a NIC scenario.

See *Configuring a NIC Scenario* on page 194.

6. Verify the NIC configuration.

See *Verifying Configuration for the NIC* on page 203.

Starting the NIC from the SRC CLI

Start the NIC component before you configure it. When you enable NIC for the first time, it creates the default operating properties for the component.

To start NIC:

- From operational mode, enable the NIC.

```
user@host> enable component nic
Starting NICHOST: may take a few minutes...
```

Reviewing and Changing Operating Properties for NIC

Before you configure a NIC configuration scenario, review the default operating properties and change values as needed. Operating properties are configured for a slot.

Reviewing the Default NIC Operating Properties

To review the default NIC operating properties:

1. From configuration mode, access the configuration statement that specifies the configuration for the NIC on a slot.

```
[edit]
user@host# edit slot number nic
```

For example:

```
[edit]
user@host# edit slot 0 nic
```

2. Run the show command.

```
[edit slot 0 nic]
user@host# show
base-dn o=umc;
java-runtime-environment ../jre/bin/java;
java-heap-size 128m;
snmp-agent;
hostname DemoHost;
initial {
    dynamic-dn "ou=dynamicConfiguration, ou=Configuration,
o=Management,<base>";
    directory-connection {
        url ldap://127.0.0.1:389/;
        backup-urls ;
        principal cn=nic,ou=Components,o=Operators,<base>;
        credentials *****;
        timeout 10;
        check-interval 60;
    }
    directory-eventing {
        eventing;
        signature-dn <base>;
        polling-interval 15;
        event-base-dn <base>;
        dispatcher-pool-size 1;
    }
    static-dn "l=OnePop,l=NIC, ou=staticConfiguration, ou=Configuration,
o=Management,<base>";
}
```

Changing NIC Operating Properties

In most cases you can use the default NIC operating properties. Change the default properties if needed for your environment.

To change NIC operating properties:

1. From configuration mode, access the configuration statement that specifies the configuration for the NIC on a slot.

```
[edit]
user@host# edit slot number nic
```

For example:

```
[edit]
user@host# edit slot 0 nic
```

2. (Optional) If you store data in the directory in a location other than the default, *o = umc*, change this value.

```
[edit slot 0 nic]
user@host# set base-dn base-dn
```

3. (Optional) If you determine that additional memory is needed, change the maximum memory size available to the (Java Runtime Environment) JRE.

```
[edit slot 0 nic]
user@host# set java-heap-size java-heap-size
```

By default, the JRE can allocate 128 MB. Set to a value lower than the available physical memory to avoid low performance because of disk swapping.

If you use an SAE plug-in agent, we recommend that you increase the JVM max heap to a value in the range 400–500 MB.

If you need help to determine the amount of memory needed, contact Juniper Networks Customer Services and Support.

4. (Optional) Enable viewing of SNMP counters through an SNMP browser.

```
[edit slot 0 nic]
user@host# set snmp-agent
```

5. (Optional) Change the name of the NIC host. Use the default name of the NIC host configured for a NIC scenario. In most cases, the NIC host name is DemoHost.

```
[edit slot 0 nic]
user@host# set hostname hostname
```

6. (Optional) Change the initial properties.

See *SRC-PE Getting Started Guide, Chapter 25, Configuring Local Properties with the SRC CLI*.

Configuring NIC Replication

You configure NIC replication to keep the NIC configuration highly available.

Before you configure NIC replication:

- Make sure that you understand how NIC groups are used.
See *Chapter 9, Locating Subscriber Information with the NIC*.
- Identify which NIC hosts are to provide redundancy for each other.
- Select a name for a group for each of these hosts.

To configure NIC replication:

1. From configuration mode, access the configuration statement that specifies the configuration for the agent.

```
[edit]
user@host# slot number nic
```

For example:

```
[edit]
user@host# slot 0 nic
```

2. Configure the runtime group for the NIC host.

```
[edit slot 0 nic]
user@host# runtime-group runtime-group
```

For example:

```
[edit slot 0 nic]
user@host# runtime-group group1
```

Configuring a NIC Scenario

The OnePop configuration scenario is the default configuration for NIC. If you want to use another configuration scenario, you first clear data for the configuration scenario and change the static DN that identifies the scenario, see *Changing NIC Configurations* on page 204.

When you select a NIC configuration scenario, the software adds the default configuration for most properties. You can modify the NIC properties, including those for agents.



CAUTION: We recommend that you change only those statements visible at the basic editing level. Contact Juniper Professional Services or Juniper Customer Support before you change any of the NIC statements not visible at the basic editing level.

To specify a NIC configuration scenario for NIC to use:

1. Make sure that the NIC component is running.

```
user@host> show component
Installed Components
Name          Version          Status
...
nic           Release: 7.0 Build: GATEWAY.A.7.0.0.0168  running
...
```

2. From configuration mode, access the statement that configures a NIC configuration scenario, and specify the name of a scenario.

```
[edit]
user@host# edit shared nic scenario name
```

For example:

```
[edit]
user@host# edit shared nic scenario OnePopLogin
```

3. View the default configuration for the configuration scenario. For example:

```
[edit shared nic scenario OnePopLogin]
user@host# show

hosts {
  DemoHost {
    configuration {
      hosted-resolvers "/realms/login/A1, /realms/login/B1,
/realms/login/C1, /realms/login/D1, /realms/ip/A1, /realms/ip/B1,
/realms/ip/C1";
      hosted-agents "/agents/LoginNameVr, /agents/VrSaeId,
/agents/IpLoginName,
/agents/PoolVr";
    }
  }
  OnePopB0 {
    configuration {
      hosted-resolvers "/realms/login/A1, /realms/login/C1, /realms/ip/A1,
/realms/ip/C1";
      hosted-agents /agents/VrSaeId;
    }
  }
  OnePopH1 {
    configuration {
      hosted-resolvers "/realms/login/B1, /realms/login/D1, /realms/ip/B1";
      hosted-agents "/agents/LoginNameVr, /agents/IpLoginName,
/agents/PoolVr";
    }
  }
}
agents {
  VrSaeId {
    configuration {
      directory {
        search-base o=Network,<base>;
        search-filter (objectclass=umcVirtualRouter);
        search-scope 2;
      }
    }
  }
}
```

```

        server-url ldap://127.0.0.1:389/;
        backup-servers-url ;
        principal cn=nic,ou=Components,o=Operators,<base>;
        credentials *****;
    }
}
}
LoginNameVr {
    configuration {
        sae-plugin {
            event-filter "(&(! (PA_USER_TYPE=INTF)) (! (PA_LOGIN_NAME=[None])))";
            number-of-events-sent-in-a-synchronization-call 50;
        }
    }
}
IpLoginName {
    configuration {
        sae-plugin {
            number-of-events-sent-in-a-synchronization-call 50;
        }
    }
}
PoolVr {
    configuration {
        directory {
            search-base o=Network,<base>;
            search-filter (objectclass=umcVirtualRouter);
            search-scope 2;
            server-url ldap://127.0.0.1:389/;
            backup-servers-url ;
            principal cn=nic,ou=Components,o=Operators,<base>;
            credentials *****;
        }
    }
}
}
}

```

4. (Optional) Update logging configuration.

See *SRC-PE Monitoring and Troubleshooting Guide, Chapter 3, Configuring Logging for SRC Components with the CLI*.

By default, NIC has the following logging enabled for a NIC host:

```

logger file-1 {
    file {
        filter !ConfigMgr,!DES,/debug;
        filename var/log/nicdebug.log;
        rollover-filename var/log/nicdebug.alt;
        maximum-file-size 10000000;
    }
}
logger file-2 {
    file {
        filter /info-;
        filename var/log/nicinfo.log;
    }
}
}

```

```
logger file-3 {
  file {
    filter /error;
    filename var/log/nicerror.log;
  }
}
```

5. For each agent that the NIC configuration scenario includes, if needed update NIC agent configuration to define properties specific to your environment, such as directory properties.

Each type of agent has different configuration properties. The output from the **show** command identifies the type of agent under the **agents** hierarchy. For example:

```
VrSaeId {
  configuration {
    directory {
      ...
    }
  }
}

LoginNameVr {
  configuration {
    sae-plugin-in {
      ...
    }
  }
}
```

For information about agent configuration, see the following sections:

- *Configuring Directory Agents* on page 197
- *Configuring SAE Plug-In Agents* on page 199

Configuring Directory Agents

Use the following configuration statements to configure NIC directory agents:

```
shared nic scenario name agents agent configuration directory {
  search-base search-base;
  search-filter search-filter;
  search-scope (0 | 1 | 2);
  server-url server-url;
  backup-servers-url backup-servers-url;
  principal principal;
  credentials credentials;
}
```

To configure a directory agent:

1. From configuration mode, access the statement that specifies the configuration for the agent.

```
[edit]
user@host# edit shared nic scenario name agents agent configuration directory
```

For example:

```
[edit]
user@host# edit shared nic scenario OnePopLogin agents VrSaeld configuration directory
```

2. Review the default configuration for the agent. For example:

```
[edit shared nic scenario OnePopLogin agents VrSaeId configuration directory]
user@host# show
search-base o=Network,<base>;
search-filter (objectclass=umcVirtualRouter);
search-scope 2;
server-url ldap://127.0.0.1:389/;
directory-backup-urls ;
principal cn=nic,ou=Components,o=Operators,<base>;
credentials *****;
```

3. (Optional) Change the distinguished name (DN) of the location in the directory from which the agent should read information.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set search-base search-base
```

For example:

```
[edit shared nic scenario OnePop agents PoolVr configuration directory]
user@host# set search-base o=myNetwork,<base>
```

You can use `<base>` in the DN to refer to the globally configured base DN.

4. (Optional) Change the directory search filter that the agent should use.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set search-filter search-filter
```

For example:

```
[edit shared nic scenario OnePop agents PoolVr configuration directory]
user@host# set search-filter objectclass=umcVirtualRouter
```

5. (Optional) Change the location in the directory relative to the base DN from which the NIC agent can retrieve information.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set search-scope (0 | 1 | 2)
```

where:

- 0—Entry specified in the `search-base` statement
- 1—Entry specified in the `search-base` statement and objects that are subordinate by one level
- 2—Subtree of entry specified in the `search-base` statement

6. For an installation on a Solaris platform, specify the location of the directory in URL string format.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set server-url ldap:// host:portNumber
```

For example, to specify the directory on a C-series platform:

```
[edit shared nic scenario OnePop agents PoolVr configuration directory]
user@host# set server-url ldap://127.0.0.1:389/
```

7. List the URLs of redundant directories. Separate URLs with semicolons.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set directory-backup-urls backup-servers-urls
```

8. Specify the DN that contains the username that the directory server uses to authenticate the NIC agent.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set principal principal
```

For example:

```
[edit shared nic scenario OnePop agents PoolVr configuration directory]
user@host# set principal cn=nic,ou=Components,o=Operators,<base>
```

9. Specify the password that the directory server uses to authenticate the NIC agent.

```
[edit shared nic scenario name agents name configuration directory]
user@host# set credentials credentials
```

10. Restart the NIC agent.

```
user@host>request nic restart agent name name
```

Configuring SAE Plug-In Agents

By default, the CORBA naming server on a C-series platform uses port 2809. The NIC host is configured to communicate with this naming server; you do not need to change JacORB properties.

Use the following configuration statements to configure NIC SAE plug-in agents:

```
shared nic scenario name agents agent configuration sae-plugin-in{
    event-filter event-filter;
    number-of-events number-of-events;
}
```

If you plan to change the event filter for the agent, make sure that you are familiar with:

- Plug-in attributes and values

See *SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Authorization and Accounting Plug-Ins with the CLI*.

- Filter syntax

See the documentation for the SAE CORBA Remote API in the SAE Core API documentation on the Juniper Networks Web site at:

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

To configure an SAE plug-in agent:

1. From configuration mode, access the statement that specifies the configuration for the agent.

```
[edit]
user@host# edit shared nic scenario name agents agent configuration sae-plug-in
```

For example:

```
[edit]
user@host# edit shared nic scenario OnePopLogin agents LoginNameVr
configuration sae plug-in
```

2. Review the default configuration for the agent. For example:

```
[edit shared nic scenario OnePopLogin agents LoginNameVr configuration sae-plug-in]
user@host# show
event-filter "(&(!(PA_USER_TYPE=INTF))(!(PA_LOGIN_NAME=[None])))";
number-of-events-sent-in-a-synchronization-call 50;
```

3. (Optional) Change an LDAP filter that change the events that the agent collects.

```
[edit shared nic scenario name agents agent configuration sae-plug-in]
user@host# set event-filter event-filter
```

Typically, you do not need to change this value. If you do want to filter other events, use the format *pluginAttribute=attributeValue* format for event filters, where:

- *pluginAttribute*—Plug-in attribute name

- *attributeValue*—Value of filter

For example:

```
[edit shared nic scenario name agents agent configuration sae-plug-in]
user@host# set event-filter PA_USER_TYPE=INTF
```

4. Specify the number of events that the SAE sends to the agent at one time during state synchronization.

```
[edit shared nic scenario name agents agent configuration sae-plug-in]
user@host# set number-of-events number-of-events
```

For example:

```
[edit shared nic scenario OnePopLogin agents LoginNameVr configuration sae plug-in]
user@host# set number-of-events 50
```

Configuring the SAE to Communicate with SAE Plug-In Agents When You Use NIC Replication

For each NIC host that uses SAE plug-in agents, configure a corresponding external plug-in for the SAE. By default, the SAE plug-in agents share events with the single SAE plug-in. You must also configure the SAE to communicate with the SAE plug-in agent in each NIC host that you use in the NIC replication.

For detailed information about NIC plug-in agents, see *Chapter 20, Reviewing the NIC Configuration*.

For information about configuring an external plug-in for the SAE, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 12, Configuring Authorization and Accounting Plug-Ins with the CLI*.

To configure an external plug-in:

1. From configuration mode, access the statement that specifies the configuration for an external plug-in for the SAE that communicates with the agent, and assign the plug-in a unique name.

[edit]

user@host# **shared sae configuration plug-ins pool name**

2. Configure CORBA object reference for the plug-in.

[shared sae configuration plug-ins pool name external]

user@host# **corba-object-reference corba-object-reference**

For the CORBA object reference, use the following syntax:

host:port-number/NameService#plugInName

where:

- *host*—IP address or name of the machine on which you installed the NIC host that supports the agent

For local host, use the IP address 127.0.0.1.

- *port-number*—Port on which the name server runs

The default port number is 2809.

- *plugInName*—Name under which the agent is registered in the naming service

Use the format *nicssae_groupname/saePort* where *groupname* is the name of the replication group. (When replication is not used, the format is *nicssae/saePort*.)

For example:

[shared sae configuration plug-ins pool name external]

user@host# **set corba-object-reference**

corbaname::127.0.0.1:2809/NameService#nicssae/saePort

3. Configure attributes that are sent to the external plug-in for a NIC host. Because the SAE plug-in agents share the event by default, you configure only one for a NIC host.

```
[shared sae configuration plug-ins pool name external]
user@host# set attr
[( router-name | user-dn | session-id | user-type | user-ip-address | login-name)]
```

Specify the plug-in options that the agent uses. You must specify the options **session-id** and **router-name**, and other options that you specified for the agent's network data types and the agent's event filter. Do not specify attributes options of the PAT_OPAQUE attribute type, such as the option **dhcp-packet**.



NOTE: Do not include attributes that are not needed.

4. Reference the NIC as a subscriber tracking plug-in.

```
[edit shared sae group name configuration plugins event-publishers]
user@host# set subscriber-tracking pool-name
```

For example, for a pool named **nic**:

```
[edit shared sae group name configuration plugins event-publishers]
user@host# set subscriber-tracking nic
```

Obtaining Interface Configuration Information for OnePopStaticRouteIp

If you use the OnePopStaticRouteIp configuration scenario, you must obtain JUNOS interface configuration information for NIC. To get this information, you must run Network Publisher on a Solaris platform to gather the interface information.

To run Network Publisher on a Solaris platform:

1. Install NIC on a Solaris platform.

See *SRC-PE Getting Started Guide, Chapter 28, Installing the SRC Software on a Solaris Platform*.

2. On the Solaris platform, edit the `/opt/UMC/nic/etc/networkPublisher/config.properties` file and run Network Publisher. When you specify the directory configuration in the file, configure the connection to the directory on a C-series platform.

See *Chapter 12, Obtaining Interface Configuration for OnePopStaticRouteIp*.

Verifying Configuration for the NIC

After you complete the NIC configuration, verify the local NIC configuration and the NIC configuration scenario information.

To verify NIC configuration:

1. In configuration mode, run the **show** command at the [edit slot 0 nic] hierarchy level.

```
[edit slot 0 nic]
user@host# show
```

2. In configuration mode, run the **show** command at the [edit shared nic scenario *name*] hierarchy level.

For example:

```
[edit shared nic scenario OnePop]
user@host# show
```

Testing a NIC Resolution

To test a NIC resolution:

- Run the **test nic resolve** command.

```
user@host> test nic resolve <locator locator> <key key>
```

where:

- *locator*—Name of locator that requests information on behalf of an application
- *key*—Value to be resolved. This value must be of the same NIC data type configured in the NIC locator.

For example:

```
user@host> test nic resolve locator /nicLocators/ip key 10.10.10.10
```

Stopping a NIC Host on a C-series Platform

If you run NIC in client/server mode, you can stop the NIC host independently of the NIC proxy.

To stop a NIC host:

- From operational mode, disable the NIC.

```
user@host> disable component nic
```

Restarting the NIC

To restart a NIC host:

- From operational mode, restart the NIC.

```
user@host> request restart nic
```

You can also restart the NIC at the slot level.

Restarting a NIC Agent

You can restart a NIC agent to have the agent read all data in the directory again. Restart a NIC agent if the agent is not synchronized with the directory, or if you switch from one directory to another.

To restart a NIC agent:

- From operational mode, restart the agent.

```
user@host> request nic restart agent name name
```

You can restart all NIC agents by omitting an agent name for the **request nic restart agent** command.

You can also restart a NIC agent at the slot level.

Restarting a NIC Resolver

In rare instances, such as when you are troubleshooting a NIC configuration, you may want to restart a NIC resolver.

To restart a NIC resolver:

- From operational mode, restart a resolver.

```
user@host> request nic restart resolver name name
```

You can restart all NIC resolvers by omitting a resolver name for the **request nic restart resolver** command.

You can also restart a NIC resolver at the slot level.

Changing NIC Configurations

If you change the type of NIC resolution that you use in your network (for example, from the OnePop configuration scenario to the OnePopAllRealms configuration scenario), delete any existing data and specify a static DN that identifies the DN for the new NIC configuration scenario; otherwise, the new NIC configuration may not perform resolutions correctly.

To change the type of NIC resolution that you use in your network:

1. Set the editing level for the CLI to expert.

```
user@host> set cli level expert
```

2. Disable the NIC:

```
user@host> disable component nic
```

3. Delete the NIC configuration data for the existing configuration scenario from the directory.

```
user@host> request nic clear scenario-data
```

4. Navigate to the [edit slot 0 nic] hierarchy level.

5. Change the value of the **static-dn** for the local configuration to identify the location of the DN for the new configuration scenario. For example:

```
[edit slot 0 nic]
user@host# set initial static-dn "I=OnePopSharedIp, I=NIC,  
ou=staticConfiguration, ou=Configuration, o=Management,<base>"
```

6. Return to operational mode, and restart the NIC host.

```
user@host>request nic slot number restart
```

7. Set the editing level for the CLI to expert.

```
user@host> set cli level basic
```

8. Configure the new NIC scenario.

Chapter 11

Configuring NIC on a Solaris Platform

This chapter describes how to configure the network information collector (NIC) on a Solaris platform using the SRC configuration applications that run only on Solaris platforms.

You can also use the CLI that runs on Solaris platforms and the C-series platform to configure the NIC. See *Chapter 10, Configuring NIC with the SRC CLI*.

Topics in this chapter include:

- Before You Configure NIC Hosts on page 208
- Configuring NIC Hosts to Resolve Requests on page 208
- Configuring Operating Parameters for NIC Hosts on page 209
- Reviewing Basic Configuration for a NIC Host on page 213
- Modifying Basic Configuration for NIC Agents on a NIC Host on page 214
- Starting NIC on a Solaris Platform on page 221
- Stopping a NIC Host on a Solaris Platform on page 221
- Monitoring NIC Hosts on page 222
- Configuring NIC Replication on page 222
- Changing NIC Configurations on page 222

Before You Configure NIC Hosts

When you use NIC in a client/server configuration, you configure the NIC hosts before you configure the NIC proxies.

Before you configure NIC hosts on a Solaris platform:

- Plan your NIC implementation:
 - Choose the NIC configuration scenario to use.
 - Make a list of the agents in the NIC configuration scenario.

For information about NIC configuration scenarios and NIC agents, see *Chapter 9, Locating Subscriber Information with the NIC*.

- Install and configure the main SRC components, such as SAEs and the directory.

See the *SDX Getting Started Guide*.

- Install the sample data.

See *SRC-PE Getting Started Guide, Chapter 29, Defining an Initial Configuration on a Solaris Platform*.

- Install the NIC software (UMCnic package) on each system that is to support a NIC host.



NOTE: A machine can support only one NIC host.

- Understand how to use SDX Configuration Editor to create new configuration files, modify configuration files, and export files to the directory.
- Ensure that the appropriate type of router initialization script is configured for the router or network device.

See *Chapter 9, Locating Subscriber Information with the NIC*.

Configuring NIC Hosts to Resolve Requests

To configure a NIC host:

1. Configure initial operating properties for NIC.

See *Configuring Operating Parameters for NIC Hosts* on page 209.

2. Review the configuration for the instance of the NIC host that you use.

See *Reviewing Basic Configuration for a NIC Host* on page 213.

3. Modify configuration for NIC agents.

See *Modifying Basic Configuration for NIC Agents on a NIC Host* on page 214.

4. Start NIC on the NIC hosts.

See *Starting NIC on a Solaris Platform* on page 221.

5. Configure NIC replication to make NIC highly available.

See *Configuring NIC Replication* on page 222.

6. If you use the OnePopStaticRouteIp configuration scenario, configure NIC to collect interface information for JUNOS routing platforms.

See *Chapter 12, Obtaining Interface Configuration for OnePopStaticRouteIp*.

Configuring Operating Parameters for NIC Hosts

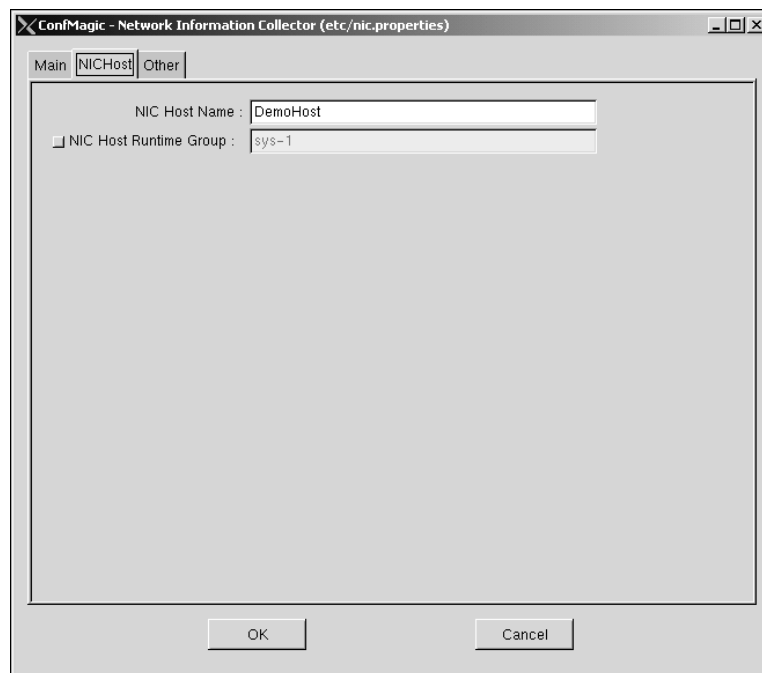
The operating parameters define how the NIC host interacts with other SRC components, such as the directory.

To configure the operating parameters:

1. Log in as root.
2. Start the local configuration tool in the directory where you installed the NIC.

/opt/UMC/nic/etc/config

The Network Information Collector window appears.



3. Configure the fields in each tab of this window. The following sections describe the properties on each tab:
 - Directory Connection Properties for NIC Hosts on page 210
 - NIC Host Properties on page 212
 - Additional Properties for NIC Hosts on page 212
4. Click OK.



NOTE: If you change any of the NIC operating parameters, restart NIC for the changes to take effect.

Directory Connection Properties for NIC Hosts

In the Main tab of the Network Information Collector window of the local configuration tool, you can modify the following fields to configure directory connection properties.

Primary Directory Server

- Location of the directory server in URL string format.
- Value—URL in the format [ldap | ldaps]:// { <host> } : <portNumber>
 - ldap—LDAP connection (not secure)
 - ldaps—Secure LDAP connection
 - <host> —Name or IP address of the host that supports the directory
 - <portNumber> —Number of the TCP/IP port

- Default—`ldap://127.0.0.1:389/`
- Example—`ldaps://192.0.2.10:389/`

Backup Directory Servers

- List of redundant directories.
- Value—List of URLs separated by semicolons.
For format of the URL, see the field Primary Directory Server on page 210.
- Example—`ldaps://192.0.2.10:389/`

Base DN

- Location in the directory in which the SRC data is stored.
- Value—DN
- Example—`o = UMC`

Bind DN

- DN that contains the username that the directory server uses to authenticate the NIC host.
- Value—`< DN > , < base >`
- Example—`cn = nic, ou = Components, o = Operators, < base >`

Bind Password

- Password that the directory server uses to authenticate the NIC host.
- Value—Text string or Base64 string
- Example—`nic`

Static Configuration DN

- DN of the location in which the NIC configuration is stored.
- Value—DN
- Example—`l = OnePop, l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc`

Dynamic Configuration DN

- DN of the location in which data that the NIC automatically generates is stored.
- Value—DN
- Example—`ou = dynamicConfiguration, ou = Configuration, o = Management, o = umc`

Connect Timeout(s)

- Time that the NIC waits for the directory server to respond when it tries to connect to the directory.
- Value—Number of seconds in the range 1–2147483647
- Example—10

NIC Host Properties

In the NIC Host tab of the Network Information Collector window of the local configuration tool, you can modify the following fields.

NIC Host Name

- Name of the NIC host that you configured.
- Value—Text string
- Guidelines—Use the name DemoHost. The configuration scenarios all use DemoHost as the NIC hostname.
- Default—No value
- Example—DemoHost

NIC Host Runtime Group

- Group to which this NIC host belongs if you configure NIC replication.
- Value—Text string
- Default—No value
- Example—ontarioHosts

Additional Properties for NIC Hosts

In the Other tab of the Network Information Collector window of the local configuration tool, you can modify the following fields.

NIC Host Java

- Path to the JRE.
- Value—Path (absolute or relative) to the directory that contains the JRE
- Example—../jre/bin

JVM Max Heap

- Maximum memory size available to the JRE.
- Value—Capacity in megabytes

- Guidelines—By default, the JRE can allocate 128 MB. Change this value if you have problems because of lack of memory. Set to a value lower than the available physical memory to avoid low performance because of disk swapping.
If you use an SAE plug-in agent, we recommend that you increase the JVM max heap to a value in the range 400–500 MB.
- Default—128 MB

Enable Sysman Clients

- Specifies whether or not there is support for viewing SNMP counters with an SNMP browser.
- Value
 - Yes—Enabled
 - No—Disabled
- Default—No

Sysman IOR

- Folder that contains the IOR file for the NIC. The NIC writes its object references to this folder, and the SNMP agent discovers NIC components by monitoring the NIC IOR file in this folder.
- Value—Path to the folder that contains the IOR
- Guidelines—By default, the NIC IOR file is in the *var* folder, which is relative to the SNMP agent installation folder (*/opt/UMC/agent*). You need to change this property only if you installed the SNMP agent in a folder other than the default folder, or if you previously changed this property and now need it to point to the folder where the IOR file currently resides.
- Default—*/opt/UMC/agent/var*

Reviewing Basic Configuration for a NIC Host

Typically, you can use the configuration for an instance of a NIC host in a configuration scenario. Review the host configuration and change the configuration if needed.

Use the configuration for the NIC host DemoHost. This configuration is intended for use with NIC replication. You can configure one NIC host on a machine.

To use Configuration Editor to review NIC host configuration for a sample scenario:

1. In the NIC folder, open the configuration file whose name matches the scenario you want to modify for your configuration.
2. Click the **Hosts** tab, and review NIC host configurations. No changes should be needed.

After you review the configuration for an instance of a NIC host, modify the NIC agent configuration.

See *Modifying Basic Configuration for NIC Agents on a NIC Host* on page 214.

Modifying Basic Configuration for NIC Agents on a NIC Host

You configure NIC agents for a NIC host by modifying the agent configuration in a configuration scenario provided in the NIC sample data. Typically, you use the configuration for the NIC host DemoHost. This configuration is intended for use with NIC replication.

To use SDX Configuration Editor to configure agents for a NIC host by modifying a sample scenario:

1. Make sure you know which type of agents your configuration scenario uses.

If you do not know which agents the scenario uses, see *Chapter 9, Locating Subscriber Information with the NIC*.

2. Update NIC agent configuration to define properties specific to your environment, such as directory properties. See the following sections:

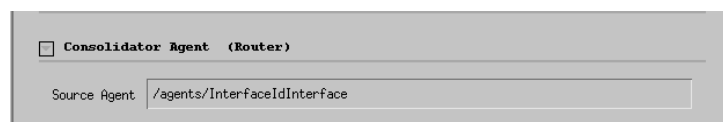
- *Configuring Consolidator Agents* on page 214
- *Configuring Directory Agents* on page 215
- *Configuring SAE Plug-In Agents* on page 217

Configuring Consolidator Agents

Use the basic editing level in SDX Configuration Editor to update the basic configuration for an agent.

To use SDX Configuration Editor to modify the configuration for consolidator agents:

1. In the navigation pane, select a configuration file for NIC.
2. Select the **Agents** tab, and expand the **Consolidator Agent** section.



3. Review the entry in the Source Agent field in the Consolidator Agent section of the Agents pane, and modify it if needed.

See *Consolidator Agent Fields in Basic Editing Level* on page 215.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP directory**.

Consolidator Agent Fields in Basic Editing Level

In SDX Configuration Editor, you can modify the following field in the Consolidator Agent section of the Agents pane in a NIC configuration file.

Source Agent

- Path to the agent for which this consolidator agent publishes data.
- Value—Text string
- Default—No value
- Example—*/agents/InterfaceIdInterface*
- Property name—sourceAgent

Configuring Directory Agents

Use the basic editing level in SDX Configuration Editor to update the basic configuration for an agent.

To use SDX Configuration Editor to modify the configuration for directory agents:

1. In the navigation pane, select a configuration file for NIC.
2. Select the **Agents** tab, and expand the **Directory Client Agent** section.

3. Review the entries in the Directory Agent section of the Agents pane, and modify if needed.

See *Directory Agent Fields in Basic Editing Level* on page 216.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP directory**.

Directory Agent Fields in Basic Editing Level

In SDX Configuration Editor, you can modify the following fields in the Directory Client Agent section of the Agents pane in a NIC configuration file.

Search Base

- DN of the location in the directory from which the agent should read information.
- Value— < DN > , < base >
- Default—No value
- Example—*o = Network, < base >*
- Property name—baseDN

Search Filter

- Directory search filter that the agent should use.
- Value—LDAP search filter
- Guidelines—Optional field.
- Default—No value
- Example—(objectclass = umcVirtualRouter)
- Property name—searchFilter

Search Scope

- Location in the directory relative to the base DN from which the NIC agent can retrieve information.
- Value—One of the following options:
 - Object—Entry specified in the Search Base field only
 - Level—Entry specified in the Search Base field and objects that are subordinate by one level
 - Subtree—Subtree of entry specified in the Search Base field
- Guidelines—Optional field.
- Default—Subtree
- Property name—searchScope

Server URL

- Location of the directory in URL string format.
- Value—Location of the directory that stores configuration information in URL string format `ldap:// <host> : <portNumber>`
 - `<host>` —IP address or name of directory host
 - `<portNumber>` —Number of TCP/IP port
- Default—No value
- Example—`ldap://127.0.0.1:389/`
- Property name—`java.naming.provider.url`

Backup Servers URL

- List of redundant directories.
- Value—List of URLs separated by semicolons
- Default—No value
- Example—`ldap://127.0.0.1:389/`
- Property name—`net.juniper.smgmt.des.backup_provider_urls`

Authentication DN

- DN that contains the username that the directory server uses to authenticate the NIC agent.
- Value—`<DN> , <base>`
- Default—No value
- Example—`cn = nic, ou = Components, o = Operators, < base >`
- Property name—`java.naming.security.principal`

Password

- Password that the directory server uses to authenticate the NIC agent.
- Value—Text string or Base64 string
- Default—No value
- Example—`nic`
- Property name—`java.naming.security.credentials`

Configuring SAE Plug-In Agents

Use the basic editing level in SDX Configuration Editor to update the basic configuration for an agent.

To use SDX Configuration Editor to modify the configuration for SAE plug-in agents:

1. In the navigation pane, select a configuration file for NIC.
2. Select the **Agents** tab, and expand the **SAE Plugin Agent** section.

3. Review the entries in the fields in the SAE Plugin Agent section of the Agents pane, and modify if needed.

See *SAE Plug-In Agent Fields in Basic Editing Level* on page 218.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP directory**.

In addition, configure the SAE for the SAE plug-in agent, see *Configuring the SAE for SAE Plug-In Agents* on page 219.

If you use NIC replication, see *Configuring the SAE to Communicate with SAE Plug-In Agents When You Use NIC Replication* on page 219.

SAE Plug-In Agent Fields in Basic Editing Level

In SDX Configuration Editor, you can modify the following fields in the SAE Plug-In Agent section of the Agents pane in a NIC configuration file.

Event Filter

- LDAP filter that restricts the events that the agent collects.
- Value— < pluginAttribute > = < attributeValue >
 - < pluginAttribute > —Plug-in attribute name
 - < attributeValue > —Value of filter
- Default—No value
- Example—PA_USER_TYPE = INTF
- Property name—eventFilter

Number of Events Sent in a Synchronization Call

- Number of events the SAE sends to the agent at one time during state synchronization.
- Value—Integer in the range 1–2147483647
- Guidelines—This field is used if state synchronization is enabled for the SAE plug-in agent. State synchronization is enabled by default.
- Default—50
- Property name—stateSyncBulkSize

Configuring the SAE for SAE Plug-In Agents

For each SAE plug-in agent in your configuration, you must also configure a corresponding external plug-in for the SAE. Use the following guidelines:

- For the CORBA object reference, use the following construction:

`<host>:<port-number>/NameService#<plugInName>`

- `<host>` —IP address or name of the machine on which you installed the NIC host that supports the agent
- `<port-number>` —Port on which the name server runs
- `<plugInName>` —Name of the agent
- Specify the plug-in attributes that the agent uses. You must specify the attributes `PA_SESSION_ID` and `PA_ROUTER_NAME`, and other attributes that you specified for the agent's network data types and the agent's event filter. Do not, however, specify attributes of type `PAT_OPAQUE`, such as the attribute `PA_DHCP_PACKET`.



NOTE: Do not include attributes that are not needed.

For information about configuring an external plug-in for the SAE, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 9, Configuring Internal, External, and Synchronization Plug-Ins with the SRC CLI*.

Configuring the SAE to Communicate with SAE Plug-In Agents When You Use NIC Replication

You must configure the SAE to communicate with each SAE plug-in agent in each NIC host that you use in the NIC replication.

To use SDX Configuration Editor to configure the SAE to communicate with an SAE plug-in agent in a NIC host:

1. In the navigation pane, select the SAE configuration file for the SAE that communicates with the agent.
2. Select the **Plug-Ins** tab, and expand the **Plug-In Pool** section.
3. Create a new instance of an external plugin, and assign the external plug-in a unique name.

4. Either configure the agent as a global-user tracking plug-in or as a retailer-specific tracking plug-in.

This action specifies which events the SAE sends to the agents.

See *Configuration Fields for SAE Plug-In Agents* on page 220.

5. Repeat Steps 3 to 4 for each SAE plug-in agent in each NIC host that you use in the NIC replication.

For information about these tasks, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 10, Overview of Configuring Plug-Ins for Solaris Platforms*.

Configuration Fields for SAE Plug-In Agents

Use the information in the following field descriptions as guidelines when configuring external plug-ins for SAE plug-in agents. You can configure these fields in SDX Configuration Editor the Plug-Ins tab of an SAE configuration file.

CORBA Object Reference

- CORBA object reference for the plug-in.
- Value—CORBA object reference in the format:
corbaname:: < host > :900/NameService# < agentName > _ < groupName > /saePort < plugInName >
 - < host > —IP address or name of the machine on which you installed the NIC host that supports the agent
 - < agentName > —Name of the agent
 - < groupName > —Name of the group to which the NIC host that supports the agent belongs
- Default—No value
- Example—corbaname::192.168.0.100:900/NameService#nicsae_sys-1/saePort
- Property name—Plugin. < pluginName > .objectref
 - < pluginName > —Name of the external plug-in that you created, such as nic1

Attributes

- Plug-in attributes that the agent uses.
- Value—Comma-separated list of plug-in attributes. For a complete list of attributes.

See *SRC-PE Subscribers and Subscriptions Guide, Chapter 11, Configuring Authorization and Accounting Plug-Ins*.
- Guidelines—You must specify the attributes PA_SESSION_ID, PA_ROUTER_NAME, and other attributes that you specified for the agent's network data types and the agent's event filter. Do not, however, specify attributes of type PAT_OPAQUE, such as the attribute PA_DHCP_PACKET. Use only the attributes that you need to lessen effect on system performance.
- Default—Comma-separated list of all possible attributes

- Example—PA_SESSION_ID, PA_ROUTER_NAME
- Property name—Plugin. <pluginName> .attr
 - <pluginName> —Name of the external plug-in that you created, such as nic1

Global User Tracking Plug-ins

- Tracks all subscriber sessions. These plug-in instances are called after a subscriber session starts and after a subscriber session ends.
- Value—Comma-separated list of plug-in instances
- Default—fileAcct
- Example—fileAcct, nic1, nic2
- Property name—User.tracking.plugins

Starting NIC on a Solaris Platform

If you run NIC in client/server mode, after you configure operating parameters for a NIC host and modify basic configuration for a NIC host, start the NIC host.

To start a NIC host:

1. On the machine on which the NIC host is installed, log in as `root` or as an authorized nonroot admin user.
2. Start the NIC host from its installation directory.

`/opt/UMC/nic/etc/nichost start`

Stopping a NIC Host on a Solaris Platform

If you run NIC in client/server mode, you can stop the NIC host independently of the NIC proxy.

To stop a NIC host:

1. On the machine on which the NIC host is installed, log in as `root` or as an authorized nonroot admin user.
2. Stop the NIC host from its installation directory.

`/opt/UMC/nic/etc/nichost stop`

Monitoring NIC Hosts

To verify that a NIC host is running:

1. On the machine on which the NIC host is installed, log in as **root** or as an authorized nonroot admin user.
2. Verify the status of the NIC host from its installation directory.

`/opt/UMC/nic/etc/nichost status`

Configuring NIC Replication

You can configure NIC replication for a new NIC configuration and for an established NIC configuration.

To configure NIC replication:

1. Assign NIC hosts to groups:
 - a. Log in as **root** on the machine on which you installed a NIC host.
 - b. Access the operating parameters, and configure the field called NIC Host Runtime group.

See [Configuring Operating Parameters for NIC Hosts](#) on page 209.

- c. If the NIC host was already running, restart it.
 - d. Repeat Steps a to c for each NIC host in the group.
2. Configure the NIC proxy to communicate with groups of NIC hosts.

See [Chapter 13, Configuring Applications to Communicate with an SAE](#).

Changing NIC Configurations

If you change the type of NIC resolution that you use in your network (for example, from the OnePop resolution to the OnePopAllRealms resolution), delete the old data; otherwise the new NIC configuration may not perform resolutions correctly.

To change the type of NIC resolution that you use in your network:

1. Delete the NIC configuration data for the old resolution from the DN *ou = dynamicConfiguration, ou = Configuration, o = Management, o = umc* in the directory.

You can delete the old data with SDX Admin or another LDAP client.

2. Configure the new NIC scenario.

Chapter 12

Obtaining Interface Configuration for OnePopStaticRouteIp

This chapter describes how to obtain configuration information for a JUNOS interface for use with the OnePopStaticRouteIp configuration scenario in NIC. Topics include:

- JUNOS Interface Information for OnePopStaticRouteIp on page 223
- Information Collection for OnePopStaticRouteIp from the Network Publisher on page 224
- Before You Run the Network Publisher on page 224
- Configuring the Network Publisher on page 224
- Running the Network Publisher on page 229
- Troubleshooting Router Connections and Configuration for the Network Publisher on page 229
- Changing the Location of an Input Directory for the Network Publisher on page 230
- Reviewing the Information Collected from a JUNOS Routing Platform on page 230
- Reviewing and Editing Interface Information from SDX Admin on page 231

JUNOS Interface Information for OnePopStaticRouteIp

The OnePopStaticRouteIp configuration scenario for NIC resolves an IP address for a subscriber whose traffic enters the network through a JUNOS interface to a reference for the SAE that manages the interface. To perform this resolution, the NIC needs information about the JUNOS interfaces. The Threat Mitigation Application Portal relies on the OnePopStaticRouteIp configuration scenario.

The interface information is stored in the directory in an XML document. You can add the interface information to the directory and update the information as needed from the network publisher. The network publisher is a NIC component that collects interface information from the routing tables on specified JUNOS routing platforms. You can view and update the interface configuration from SDX Admin.

Information Collection for OnePopStaticRoutelp from the Network Publisher

The network publisher gathers information about interfaces on specified JUNOS routers and then stores that information in the directory. You run the network publisher whenever you want to get interface information from one or more routers; NIC does not automatically update configuration information in the directory.

The network publisher uses a configuration file on the system on which the NIC host is configured. When you run the network publisher, the information in the configuration file determines the types of information to be collected.

Before You Run the Network Publisher

When you run the network publisher, it connects to a number of JUNOS routing platforms through Telnet.

Before you run the network publisher:

- Verify the version of the JUNOS software that is running on each JUNOS routing platform.

Typically, all of the JUNOS routing platforms should run the same version of the JUNOS software.

- Make sure that a Telnet service is enabled on each router from which the network publisher is to collect interface information.

Configuring the Network Publisher

To configure the network publisher:

- In a text editor, edit the file */opt/UMC/nic/etc/networkPublisher/config.properties*.

For information about the fields in this file, see *Network Publisher Configuration File Fields* on page 224.

Network Publisher Configuration File Fields

The */opt/UMC/nic/etc/networkPublisher/config.properties* file contains the following types of configuration fields:

- *Logging Configuration Fields* on page 225
- *Router Configuration Fields* on page 225
- *Filter Configuration Fields* on page 226
- *Directory Configuration Fields* on page 227
- *Troubleshooting Configuration Fields* on page 228

Entries in the file have the format `< field > = < value > .`

The network publisher identifies routers by a number; for example, r1, r2.

Logging Configuration Fields

The network publisher uses the same logging properties as other SRC components.

For information about logging properties and about managing log files, see:

- *SRC-PE Monitoring and Troubleshooting Guide, Chapter 4, Configuring Logging for SRC Components on a Solaris Platform*

Router Configuration Fields

The router configuration fields provide information about the JUNOS routing platforms on which interfaces reside for which the network publisher collects information.

`/routers/tags.release`

- Release number of the JUNOS software installed on the JUNOS routing platform.
- Value—Release number in the format:
`< major release > . < minor release > R < revision >`
- Guidelines—If a value is also specified under `/routers/r < number > tags.release`, the value for the specified router is used for that router.
- Default—No value
- Example—`/routers/tags.release = 7.6R1`

`/routers/tags.hostname`

- Hostname of the machine on which the network publisher runs.
- Value—`< hostname >`
- Default—No value
- Example—`/routers/tags.hostname = myhost`

`/routers/junoscript-authentication.username`

- Username to log in to the JUNOScript server.
- Value—`< username >`
- Default—No value
- Example—`/routers/junoscript-authentication.username = root`

/routers/junoscript-authentication.challenge_response

- Password to log in to the JUNOScript server
- Value— < password >
- Default—No value
- Example—/routers/junoscript-authentication.challenge_response = secret

/routers/r<number>/hostname

- Hostname of a JUNOS routing platform.
- Value— < hostname >
- Default—No value
- Example—/routers/r1/hostname = RouterExternal

/routers/r<number>/address

- IP address of the JUNOS routing platform for which you specified a hostname.
- Value— < IP address in dotted decimal notation >
- Default—No value
- Example—/routers/r1/address = 10.10.10.10

/routers/r<number>/tags.release

- Release number of the JUNOS software installed on a specific JUNOS routing platform.
- Value—Release number in the format:
 < major release > . < minor release > R < release number >
- Guidelines—If a value is also specified under */routers/tags.release*, the value for the specified router is used for that router.
- Default—No value
- Example—/routers/r1/tags.release = 7.6R1

Filter Configuration Fields

The filter configuration fields specify filters that the network publisher uses to collect information from JUNOS routing platforms. You can specify two filters.

/transform/route_table_filter

- Routing table from which the network publisher collects interface information.
- Value—Element name in the format:
 (element-name = < value >)
- Default—(table-name = inet.0)
- Example—/transform/route_table_filter = (table-name = inet.0)

/transform/route_entry_filter

- Element(s) in a specified router table from which the network publisher collects interface information.
- Value—Element name in the format:
(< element-name > = < value >)
- Default—(protocol-name = *)
- Example—/transform/route_entry_filter = (protocol-name = *)

Directory Configuration Fields

The directory fields specify information used to connect to the directory.

/dir/java.naming.provider.url

- URL of the primary directory.
- Value—URL in the format ldap:// < host > :389
< host > —IP address or name of directory host
- Default—No value
- Example—/dir/java.naming.provider.url = ldap://127.0.0.1:389/

/dir/java.naming.security.principal

- Distinguished name (DN) of the directory entry that defines the username with which the network publisher accesses the directory.
- Value— < DN >
- Default—No value
- Example—/dir/java.naming.security.principal = cn = umcadmin, o = umc

/dir/java.naming.security.credentials

- Password with which the network publisher accesses the directory.
- Value— < password >
- Guidelines—The password can be encoded in base64 and not visible in plain text. To use an encoded value, use the format {BASE64} < encoded-value > .
- Default—No value
- Example
 - /dir/java.naming.security.credentials = admin123
 - /dir/java.naming.security.credentials = {BASE64}c3Nw

/dir/baseDN

- Subtree in the directory that stores router data.
- Value— < DN >
- Default—o = Network, o = umc
- Example—/dir/baseDN = o = Network, o = umc

Troubleshooting Configuration Fields

The troubleshooting configuration fields let you specify file information that you can use to troubleshoot configuration for the network publisher.

/routers/r<number>/session_type

- File that contains properties for the routers.
- Value— < filename >
- Guidelines—By default, the network publisher obtains router information through a Telnet session. You can specify an input file for one or more routers to troubleshoot the configuration for the network publisher. The values in an input file for a specified router take precedence over values obtained from the router through a Telnet session.
- Default—LocalFile
- Example—/routers/r1/session_type = LocalFile

/routers/r<number>/input_dir

- Directory that contains the < router_name > _1.xml document (where router_name is the hostname of a router), which contains router properties for the network publisher.
- Value— < directory-name >
- Guidelines—Use a file in the input directory if you do not want to connect to the router to obtain the interface configuration information. Use a file defined by routers/r < number > /session_type in this directory to troubleshoot the configuration for the network publisher.
- Default—/opt/UMC/nic/sample/junos/rt
- Example—/routers/r1/input_dir = /opt/UMC/nic/myconfig

/routers/r<number>/output_dir

- Directory that contains the < router_name > _1.xml document (where router_name is the hostname of a router), which contains interface configuration information collected from the routing table on a JUNOS routing platform.
- Value— < directory-name >
- Guidelines—You must specify an output directory for information to be written to an output file. You can read the information stored in files in this directory to determine whether they contain the expected information from the routing table on the specified JUNOS routing platform.
- Default—opt/UMC/nic/var/junos/rt
- Example—/routers/r1/output_dir = /var/junos/mydir

Running the Network Publisher

You run the network publisher each time you want to collect information about interfaces on JUNOS routing platforms.

To run the network publisher:

1. Move to the *etc* directory that is under the NIC installation directory, typically */opt/UMC/nic/etc*.
2. Run the **networkPublisher appl** command:

```
./networkPublisher appl
```

Troubleshooting Router Connections and Configuration for the Network Publisher

You can troubleshoot the connection between the network publisher and one or more routers, and the configuration on the routers by providing configuration information to the publisher from a file rather than from JUNOS routing platforms.

To specify that the network publisher obtain router configuration information from a file:

1. Edit the */opt/UMC/nic/etc/networkPublisher/config.properties* file.
2. Specify an input directory for the routers by configuring the following property in the file:

```
/routers/r<number>/input_dir =<directory>
```

For example:

```
/routers/r1/input_dir =/opt/UMC/nic/myconfig
```

3. Specify that the network publisher obtain properties for a router from a file on the local system, instead of from the directory, by configuring the following property in the file:

```
/routers/r<number>/session_type = LocalFile
```

For example:

```
/routers/r1/session_type=LocalFile
```

By default, the input file is */opt/UMC/nic/sample/junos/rt/<router_name>_1.xml* where *router_name* is the hostname of a JUNOS routing platform.

You can change the location of the input directory.

See *Changing the Location of an Input Directory for the Network Publisher* on page 230.

Changing the Location of an Input Directory for the Network Publisher

By default, an input file is located in the directory `/opt/UMC/nic/sample/junos/rt`. You can use an input file to troubleshoot the configuration of the network publisher.

You can define a different location for this file in the network publisher configuration in the `/opt/UMC/nic/etc/networkPublisher/config.properties` file.

To change the location of an input file that contains router configuration:

- Edit the `config.properties` file. Specify that the network publisher locate an input file by configuring the following property in the file:

```
/routers/r<number>/input_dir =
```

See *Network Publisher Configuration File Fields* on page 224.

Reviewing the Information Collected from a JUNOS Routing Platform

To review information that the network publisher collects from a JUNOS routing platform:

1. Locate the directory that contains the output file(s) by reviewing the value of the following properties in the `/opt/UMC/nic/etc/networkPublisher/config.properties` configuration file:

- For one router:

```
/routers/r1/output_dir =
```

- For all routers:

```
/routers/output_dir =
```

2. In the directory specified `/routers/r1/output_dir`, open the `<router_name>_1.xml` document (where `router_name` is the hostname of a JUNOS routing platform), and review the file content.

If the information in the file is different from the information expected, there may be a problem with the configuration on the router.

Reviewing and Editing Interface Information from SDX Admin

You can use SDX Admin to edit the XML document that contains interface information used by the OnePopStaticRouteIp configuration scenario for NIC. You use the network publisher to collect the interface information and populate the XML document in the directory for you.

To use SDX Admin to edit the XML document that provides interface information for JUNOS routing platforms:

1. In the navigation pane, select a router under *o = network, o = umc*.
2. Click the **Interface Configuration** tab, and edit values in the content pane.

For information about file syntax and a sample file, see *NIC Document That Maps Subscriber IP Addresses to a JUNOS Interface* on page 231.

NIC Document That Maps Subscriber IP Addresses to a JUNOS Interface

NIC stores information about subscriber IP addresses that map to JUNOS interfaces on which the associated traffic enters the network. This XML document is stored in the specified directory. These files comply with the syntax in the file */opt/UMC/nic/etc/networkConfig.xsd*. An example file */opt/UMC/nic/networkConfig.xml* shows the type of information generated by the network publisher.

For an example configuration in SDX Admin, see the Interface Configuration tab for *o = OrderedCimKeys = THMA1, o = network, o = umc*.

Chapter 13

Configuring Applications to Communicate with an SAE

This chapter provides information about configuring NIC proxies. Topics include:

- Overview of NIC Proxy Configuration on page 233
- Before You Configure a NIC Proxy on page 233

Overview of NIC Proxy Configuration

You configure applications to communicate with network information collector (NIC) hosts. A NIC host can be local within an application, or external to the application. For Java applications, you also configure NIC proxies as part of an application.

For a number of SRC components, such as the SRC Volume-Tracking Application (SRC-VTA) and the Dynamic Service Activator, you can configure the NIC proxy for the application from the SRC CLI. On a Solaris platform, you can also configure NIC proxies from the SDX Configuration Editor. For other applications, such as the sample residential portal, you configure the NIC proxy in a property file. If you configure a NIC proxy from a property file, the fields are the same as the fields that appear at the CLI or in SDX Configuration Editor. When you develop and test SRC components that use a NIC, you can configure a NIC proxy stub to take the place of the NIC host.

For more information about NIC proxies, see *Chapter 9, Locating Subscriber Information with the NIC*.

Before You Configure a NIC Proxy

The values that you configure for a NIC proxy depend on the particular application; for example, you must specify the type of data used for the key and the type of data used for the value for each application.

Before you configure a NIC proxy for an application, obtain the following information from the system manager who maintains the NIC configuration for NIC hosts:

- The name of the resolver that the application uses.
- The type of key the application will provide to the NIC host.
- The type of value the NIC host is to return.
- Whether or not the application will use a local NIC host.
- If the application does not use a local NIC host:
 - The size of the NIC proxy cache.
 - The groups to be listed for NIC host selection. These groups provide NIC replication.

Also, if you use the SRC software on a Solaris platform and use a Java Runtime Environment (JRE) other than the one included in the SRC software distribution, review the configuration for the object request broker (ORB) to ensure that it meets the requirements for the NIC.

For information about the ORB, see *Chapter 15, Configuring Applications to Communicate with an SAE with SDX Configuration Editor*.

Chapter 14

Using the CLI to Configure SRC Applications to Communicate with an SAE

You can use the CLI to configure SRC applications to communicate with network information collector (NIC) hosts. This chapter describes how to configure a NIC proxy from the SRC CLI on a C-series platform or on a Solaris platform running the SRC software. Topics include:

- Configuration Statements for NIC Proxies on page 235
- Before You Configure a NIC Proxy on page 236
- Configuring Resolution Information for a NIC Proxy on page 237
- Changing the Configuration for the NIC Proxy Cache on page 238
- Configuring a NIC Proxy for NIC Replication on page 239
- Configuring NIC Test Data from the SRC CLI on page 241

Configuration Statements for NIC Proxies

Use the following configuration statements to configure a NIC proxy for the SAE from the [edit] hierarchy level.

```
shared sae configuration nic-proxy-configuration name {  
}
```

```
shared sae configuration nic-proxy-configuration name resolution {  
  resolver-name resolver-name;  
  key-type key-type;  
  value-type value-type;  
  expect-multiple-values;  
  constraints constraints;  
}
```

```

shared sae configuration nic-proxy-configuration name cache {
    cache-size cache-size;
    cache-cleanup-interval cache-cleanup-interval;
    cache-entry-age cache-entry-age;
}

shared sae configuration nic-proxy-configuration name nic-host-selection {
    groups groups;
    selection-criteria (roundRobin | randomPick | priorityList);
}

shared sae configuration nic-proxy-configuration name nic-host-selection blacklisting {
    try-next-system-on-error;
    number-of-retries-before-blacklisting number-of-retries-before-blacklisting;
    blacklist-retry-interval blacklist-retry-interval;
}

```

To configure a NIC proxy stub for SAE, use the following statements:

```

shared sae configuration nic-proxy-configuration name test-nic-bindings {
    use-test-bindings;
}

shared sae configuration nic-proxy-configuration name test-nic-bindings key-values
name {
    value;
}

```

Before You Configure a NIC Proxy

Before you configure a NIC proxy, you should have a good understanding of:

- NIC resolution
- NIC data types
- How NIC proxies work

See *Chapter 9, Locating Subscriber Information with the NIC*, *Chapter 18, NIC Resolution Process*, and *Chapter 13, Configuring Applications to Communicate with an SAE*.



NOTE: You cannot configure a local NIC host when the NIC is running on a C-series platform.

Configuring Resolution Information for a NIC Proxy

Use the following configuration statements to configure a NIC proxy:

```
shared sae configuration nic-proxy-configuration name {
}

shared sae configuration nic-proxy-configuration name resolution {
    resolver-name resolver-name;
    key-type key-type;
    value-type value-type;
    expect-multiple-values;
    constraints constraints;
}
```

To configure resolution information for a NIC proxy.

1. From configuration mode, access the configuration statement that specifies the NIC proxy configuration.

```
[edit]
user@host# edit shared sae configuration nic-proxy-configuration name
resolution
```

For example:

```
[edit]
user@host# edit shared sae configuration nic-proxy-configuration ip resolution
```

2. Specify the NIC resolver that this NIC proxy uses.

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set resolver-name resolver-name
```

This resolver must be the same as one that is configured on the NIC host. For example:

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set resolver-name /realms/ip/A1
```

3. Specify the NIC data type that the key provides for the NIC resolution.

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set key-type key-type
```

For example:

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set key-type ip
```

To qualify data types, enter a qualifier within parentheses after the data type; for example, to specify username as a qualifier for the key LoginName:

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set key-type LoginName (username)
```

- Specify the type of value to be returned in the resolution for the application that uses the NIC proxy.

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set value-type value-type
```

For example:

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set value-type Saeld
```

- (Optional) If the key can have more than one value, specify that the key can have multiple corresponding values.

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set expect-multiple-values
```

- (Optional. Available at the Advanced editing level.) If the application provides a constraint in the resolution request, specify the data type for the constraint. The constraint represents a condition that must or may be satisfied before the next stage of the resolution process can proceed.

```
[edit shared sae configuration nic-proxy-configuration ip resolution]
user@host# set constraints constraints
```

Changing the Configuration for the NIC Proxy Cache

You can modify cache properties for the NIC proxy to optimize the resolution performance for your network configuration and system resources. Typically, you can use the default settings for the cache properties. The configuration statements are available at the Advanced editing level.

Use the following configuration statements to change values for the NIC proxy cache:

```
shared sae configuration nic-proxy-configuration name cache {
  cache-size cache-size;
  cache-cleanup-interval cache-cleanup-interval;
  cache-entry-age cache-entry-age;
}
```

To configure the cache for a NIC proxy:

- From configuration mode, access the configuration statement that specifies the NIC proxy configuration. For example:

```
[edit]
user@host# edit shared sae configuration nic-proxy-configuration ip cache
```

2. Specify the maximum number of keys for which the NIC proxy retains data.

```
[edit shared sae configuration nic-proxy-configuration ip cache]
user@host# set cache-size cache-size
```

If you decrease the cache size or disable the cache while the NIC proxy is running, the NIC proxy removes entries in order of descending age until the cache size meets the new limit.

3. Specify the time interval at which the NIC proxy removes expired entries from its cache.

```
[edit shared sae configuration nic-proxy-configuration ip cache]
user@host# set cache-cleanup-interval cache-cleanup-interval
```

4. Specify the how long an entry remains in the cache.

```
[edit shared sae configuration nic-proxy-configuration ip cache]
user@host# set cache-entry-age cache-entry-age
```

Configuring a NIC Proxy for NIC Replication

Typically, you configure NIC replication to keep the NIC highly available. You configure NIC host selection to specify the groups of NIC hosts to be contacted to resolve a request, and to define how the NIC proxy handles NIC hosts that the proxy is unable to contact. The configuration statements are available at the Advanced editing level.

Use the following configuration statements to configure NIC host selection for a NIC proxy:

```
shared sae configuration nic-proxy-configuration name nic-host-selection {
  groups groups;
  selection-criteria (roundRobin | randomPick | priorityList);
}
```

```
shared sae configuration nic-proxy-configuration name nic-host-selection blacklisting {
  try-next-system-on-error;
  number-of-retries-before-blacklisting number-of-retries-before-blacklisting;
  blacklist-retry-interval blacklist-retry-interval;
}
```

To configure a NIC proxy to use NIC replication:

1. From configuration mode, access the configuration statement that specifies the NIC proxy configuration. For example:

```
[edit]
user@host# edit shared sae configuration nic-proxy-configuration ip
```

2. Specify the list of groups of NIC hosts that the NIC proxy can contact for resolution requests. Use commas to separate the group names.

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection]
user@host# set groups groups
```

For example

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection]
user@host# set groups [group1 group2]
```

3. If you configure more than one group, specify the selection criteria that the NIC proxy uses to determine which NIC host to contact.

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection]
user@host# set selection-criteria (roundRobin | randomPick | priorityList)
```

where:

- roundRobin—NIC proxy selects NIC hosts in a fixed, cyclic order. The NIC proxy always selects the next host in the list.
- randomPick—NIC proxy selects NIC hosts randomly from the list.
- priorityList—NIC proxy selects NIC hosts according to their assigned priorities in the list. If the host with the highest priority in the list is not available, the NIC proxy tries the host with the next-highest priority, and so on.

Priorities are defined by the order in which you specify the groups. You can change the order of NIC hosts in the list by using the **insert** command.

4. Access the configuration statement that specifies the NIC proxy configuration for blacklisting—the process of handling nonresponsive NIC hosts.

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection]
user@host# edit blacklisting
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection
blacklisting]
```

5. Specify whether or not the NIC proxy should contact the next specified NIC host if a NIC host is determined to be unavailable.

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection
blacklisting]
user@host# set try-next-system-on-error
```

6. (Optional) Change the number of times the NIC proxy tries to communicate with a NIC host before the NIC proxy stops communicating with the NIC host for a period of time. The default is 3.

```
[edit shared sae configuration nic-proxy-configuration ip nic-host-selection
blacklisting]
user@host# set number-of-retries-before-blacklisting
number-of-retries-before-blacklisting
```

7. (Optional) Change the interval at which the NIC proxy attempts to connect to an unavailable NIC host. The default is 15 seconds.

```
[edit shared sae configuration nic-proxy-configuration name nic-host-selection
blacklisting]
user@host# set blacklist-retry-interval blacklist-retry-interval
```

Configuring NIC Test Data from the SRC CLI

To test a resolution without NIC, you can configure a NIC proxy stub to take the place of the NIC. The NIC proxy stub comprises a set of explicit mappings of data keys and values in the NIC proxy configuration. When the SAE (or another SRC component configured to use a NIC proxy stub) passes a specified key to the NIC proxy stub, the NIC proxy stub returns the corresponding value. When you use a NIC proxy stub, no NIC infrastructure is required.

For example, you can specify a subscriber's IP address that is associated with a particular SAE. When the SRC component passes this IP address to the NIC proxy stub, the NIC proxy stub returns the corresponding SAE.

To use the NIC proxy stub for the SAE:

1. In configuration mode, navigate to the NIC proxy configuration and specify the type of key you want to map to a value.

```
[edit shared sae configuration nic-proxy-configuration name]
user@host# set resolution key-type key-type
```

For example, to specify the key ip for the ip NIC proxy configuration:

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set resolution key-type ip
```

2. Enable a NIC proxy stub for a resolution.

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set test-nic-bindings user-test-bindings
```

3. Specify the values of the keys for testing. These statements are available at the Expert CLI editing level.

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set test-nic-bindings key-values name value
```

where:

- *name*—Indicates the NIC data value for the proxy.
- *value*—Specifies a value for the NIC data type.

For example, to set up a login name to IP mapping for login name `jane@virneo.com` to the IP address `192.0.2.30`:

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set test-nic-bindings key-values jane@virneo.com 192.0.2.30
```

For example, to set up an IP to SAE ID mapping for IP address `190.0.2.30` to SAE ID identified by the URL for the CORBA IOR `corbaloc::10.227.7.145:8801/SAE`:

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set test-nic-bindings key-values 192.0.2.30
corbaloc::10.20.7.145:8801/SAE
```



NOTE: The SAE writes the value of the CORBA IOR to the `var/run` directory. The IP address in the `corbaloc` URL can be adjusted to the IP address or DNS name of the SAE.

You can use the key `ANY-KEY` to match any key for any key type. For example, if you want all IP addresses to resolve to the same SAE:

```
[edit shared sae configuration nic-proxy-configuration ip]
user@host# set test-nic-bindings key-values ANY-KEY
corbaloc::10.20.7.145:8801/SAE
```


Chapter 15

Configuring Applications to Communicate with an SAE with SDX Configuration Editor

This chapter discusses how to use SDX Configuration Editor to configure a NIC local host and NIC proxies for an SDX application. Topics Include:

The chapter contains the following sections:

- Configuring an Application to Use a Local NIC Host on page 243
- Configuring NIC Proxies from SDX Configuration Editor on page 245
- Reviewing and Updating the ORB Configuration for Applications That Include a NIC Proxy on page 251
- Testing Applications by Using a NIC Proxy Stub on page 253
- Monitoring NIC Proxies on page 258

Configuring an Application to Use a Local NIC Host

If you want an application to use a local NIC host, you enable the local NIC host and configure associated properties.

To use SDX Configuration Editor to configure a NIC host that runs within the same application:

1. In the navigation pane, select a configuration file for the NIC proxy you want to configure.
2. Depending on the configuration file, select the **NIC proxy** tab or the **NIC proxy** section on another tab.

- Expand the **Local NIC Host Configuration** section.

The screenshot shows a configuration window titled "Local NicHost Configuration". It contains three main fields:

- "Run NIC Host locally": A dropdown menu currently showing "Yes".
- "Local NIC Host Name": A text input field containing the text "DemoHost".
- "NIC static configuration DN": A text input field followed by an "Enable" button.

- In the Local NIC Host Configuration section, edit or accept the default values for the fields.

See *Local NIC Host Configuration Fields* on page 244.

- Select **File > Save**.
- Right-click the configuration file, and select **System Configuration > Export to LDAP Directory**.

For more information about local NIC hosts, see *Chapter 9, Locating Subscriber Information with the NIC*.

Local NIC Host Configuration Fields

In SDX Configuration Editor, you can modify the following fields in the NIC Host Configuration section of a NIC Proxy pane or section of a pane in a configuration file.

Run NIC Host Locally

- Specifies whether or not the NIC host is to run in the same application as NIC proxies.
- Value
 - Yes—NIC host will run in the same applications as a specified NIC proxy, or all NIC proxies.
 - No—NIC host will not run in the same application as NIC proxies.
- Default—No
- Property name—nic.localhost

Local NIC Host Name

- Name of the NIC host that is to run within the same application as the NIC proxy.
- Value— < Local NIC host name >
- Default—Name of the local NIC host; typically, DemoHost
- Property name—nic.hostname

NIC Static Configuration DN

- DN of the location in which the NIC configuration is stored.
- Value—DN
- Example—*l = OnePop, l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc*
- Property name—`nic.staticConfigDN`

Configuring NIC Proxies from SDX Configuration Editor

Before you configure NIC proxies, see *Chapter 13, Configuring Applications to Communicate with an SAE*.

Tasks to configure a NIC proxy from SDX Configuration Editor are:

- Configuring Resolution Information for a NIC Proxy on page 245.
- For applications that use client/server mode (not NIC local host mode):
 - (Optional) Configuring the NIC Proxy Cache on page 247.
 - (Optional but recommended) Configuring the NIC Proxy for NIC Replication on page 249.

Configuring Resolution Information for a NIC Proxy

To use SDX Configuration Editor to configure a NIC resolution:

1. In the navigation pane, select a configuration file for the NIC proxy you want to configure.
2. Depending on the configuration file, select the **NIC proxy** tab or the **NIC proxy** section on another tab.
3. Expand the **Resolution** section.

The screenshot shows a configuration window with a tab labeled "Resolution". Below the tab, there are several input fields:

- Resolver Name:** /realms/ip/R1
- Key Type:** Subscriber's IP address
- Value Type:** SAE server ID
- Expect Multiple Values:** No
- Constraints:** (empty field)
- Use Local NIC Host:** (checkbox, currently unchecked)

4. In the Resolution section, edit or accept the default values for the fields.

See *NIC Proxy Resolution Fields* on page 246.

5. Select **File > Save**.
6. Right-click the configuration file, and select **System Configuration > Export to LDAP Directory**.

NIC Proxy Resolution Fields

In SDX Configuration Editor, you can modify the following fields in the Resolution section of a NIC Proxy pane or section of a pane in a configuration file.

Resolver Name

- NIC resolver that this NIC proxy uses.
- Value—Path to NIC resolvers relative to the static configuration object in the directory
- Guidelines—This resolver must be the same as one that is configured on the NIC host.
- Default—No value
- Property—nic.server

Key Type

- Type of data used that the key provides for the NIC resolution.
- Value—Varies according to the application that is using the NIC proxy; choose from the menu.
- Guidelines—This key must be the same as the one for the specified resolver that is configured on the NIC host.
- Default—No value
- Property—nic.keytype

Value Type

- Type of value to be returned in the resolution.
- Value—Varies according to the application that is using the NIC proxy; choose from the menu.
- Default—No value
- Property—nic.valuetype

Expect Multiple Values

- Specifies whether or not the key can have multiple corresponding values.
- Value
 - Yes—Key can have multiple corresponding values.
 - No—Key can have only one value.
- Default—No
- Property—nic.expectmultiple

Constraints

- Condition that must or may be satisfied before the next stage of the resolution process can proceed.
- Value—Comma-separated list of the data types of constraints specified for the NIC resolution on the NIC host
- Guidelines—Provide a value only if the constraint will be provided by the application in the resolution request. Typically, you do not need to provide a value for this field.
- Default—No value
- Property—`nic.constraints`

Use Local NIC Host

- Specifies whether or not the NIC proxy uses a NIC local host if one is configured and running within the application.
- Value
 - Yes—NIC hosts runs within the same application as the NIC proxies for the application.
 - No—NIC host can run on a machine other than the one on which the NIC proxies are located.
- Default—Yes
- Property—`nic.localhost`

Configuring the NIC Proxy Cache

You can modify cache properties for the NIC proxy to optimize the resolution performance for your network configuration and system resources. Typically, you can use the default values for the cache properties.

To use SDX Configuration Editor to configure NIC proxy cache properties:

1. In the navigation pane, select a configuration file for the NIC proxy you want to configure.
2. Depending on the configuration file, select the **NIC proxy** tab or the **NIC proxy** section on another tab.
3. Expand the **Cache** section.

The screenshot shows a configuration window with a tab labeled 'Cache'. Below the tab are three rows of configuration fields:

Field Name	Field Type	Action
Cache Size	Text input	Disable
Cache Cleanup Interval	Text input	
Cache Entry Age	Text input	Disable

4. In the Cache section, edit or accept the default values for the fields.

See *NIC Proxy Cache Fields* on page 248.

5. Select **File > Save**.
6. Right-click the configuration file, and select **System Configuration > Export to LDAP Directory**.

NIC Proxy Cache Fields

In SDX Configuration Editor, you can modify the following fields in the Cache section of a NIC Proxy pane or section of a pane in a configuration file.

Cache Size

- Maximum size of the cache in which the NIC proxy retains data.
- Value—Integer in the range 0–2147483647
 - 0—Cache is disabled
 - Other values—Actual size of the cache
- Guidelines—You can change this value without restarting the NIC host. If you decrease the cache size or disable the cache while the NIC proxy is running, the NIC proxy removes entries in order of descending age until the cache size meets the new limit.
- Default—10000
- Property name—`nic.maxCacheSize`

Cache Cleanup Interval

- Time interval at which the NIC proxy removes expired entries from its cache.
- Value—Number of seconds in the range 5–2147483
- Default—15 seconds
- Property name—`nic.cleanupInterval`

Cache Entry Age

- Maximum time that the NIC proxy can cache an entry. The NIC proxy compares this property with the life expectancy of each entry and uses the lower value to determine when to remove the entry.
- Value—Number of seconds in the range 0–4294967295
 - 0 or unspecified—Life expectancy of the data, which determines expiration of data
 - Other values—Actual time that the NIC proxy caches entries
- Default—0
- Property name—`nic.maxCacheEntryAge`

Configuring the NIC Proxy for NIC Replication

In most cases, NIC hosts should be configured to use NIC replication. In the NIC Host Selection section of the pane, specify the groups of NIC hosts to be contacted to resolve a request.

To use SDX Configuration Editor to configure NIC replication for a NIC proxy:

1. In the navigation pane, select a configuration file for the NIC proxy you want to configure.
2. Depending on the configuration file, select the **NIC proxy** tab or the **NIC proxy** section on another tab.
3. Expand the **NIC Host Selection** section and the **Blacklisting** section.

The screenshot shows a configuration pane with two main sections. The top section is titled 'NICHost Selection' and contains a 'Groups' text field with a 'Disable' button to its right, and a 'Selection Criteria' text field with a checkmark icon to its right. The bottom section is titled 'Blacklisting' and contains three text fields: 'Try Next System on Error' with a checkmark icon, 'Number of Retries Before Blacklisting', and 'Blacklist Retry Interval'.

4. In the NIC Host Selection section and the Blacklisting section, edit or accept the default values for the fields.

See *NIC Host Selection Fields* on page 249.

5. Select **File > Save**.
6. Right-click the configuration file, and select **System Configuration > Export to LDAP Directory**.

NIC Host Selection Fields

In SDX Configuration Editor, you can modify the following fields in the NIC Host Selection section of a NIC Proxy pane or section of a pane in a configuration file.

Groups

- List of groups of NIC hosts that the NIC proxy can contact.
- Value—Comma-separated list of names of groups
- Default—No value
- Example—ontarioHost, vancouverHost
- Property name—nic.groups

Selection Criteria

- Selection algorithm that the NIC proxy uses to determine which NIC host to contact.
- Value
 - Round Robin—NIC proxy selects NIC hosts in a fixed, cyclic order. The NIC proxy always selects the next host in the list.
 - Random Pick—NIC proxy selects NIC hosts randomly from the list.
 - Priority List—NIC proxy selects NIC hosts according to their assigned priorities in the list. If the host with the highest priority in the list is not available, the NIC proxy tries the host with the next-highest priority, and so on.
- Guidelines—Use Round Robin or Random Pick to distribute resolution requests among NIC hosts. Use Priority List if you prefer to use a particular NIC host; for example, you may reduce operating costs by using a local NIC host.
- Default—Round Robin
- Property name—`nic.repStrategy`

Try Next System on Error

- Specifies whether or not the NIC proxy should contact the next specified NIC host if a NIC host is determined to be unavailable.
- Value
 - Yes—NIC proxy should contact the next specified host.
 - No—NIC proxy should not contact the next specified host.
- Default—Yes
- Property name—`nic.tryNextOnError`

Number of Retries Before Blacklisting

- Number of times the NIC proxy tries to communicate with a NIC host before the NIC proxy stops communicating with the NIC host for a period of time.
- Value—Integer in the range 0–2147483647
- Default—3
- Property name—`nic.numOfRetries`

Blacklist Retry Interval

- Interval at which the NIC proxy attempts to connect to an unavailable NIC host.
- Value—Number of seconds in the range 0–2147483647
- Default—15
- Property name—`nic.retryInterval`

Reviewing and Updating the ORB Configuration for Applications That Include a NIC Proxy

The JRE package (UMCjre) included in the SRC software distribution is preconfigured with JacORB.

JacORB meets the requirements for applications that include a NIC proxy. If you use a different JRE, you must ensure that it is configured with an ORB that supports value types with the Object Management Group's CORBA 2.6 standard.

If the default Java Virtual Machine (JVM) for the Web application server is UMCjre or another environment that complies with this standard, you do not need to configure the ORB. However, if this is not the case, you must configure the ORB to enable your application to communicate with the NIC.

Depending on the type of application, you can do one of the following:

- Configuring JacORB as the Default ORB on page 251
- Configuring One Web Application to Use JacORB on page 252
- Configuring a Web Application Server to Use JacORB on page 253

In this case, all Web applications, but not other Java applications, inside the Web application server will use this ORB.

For additional information about JacORB, see:

<http://www.jacorb.org/documentation.html>

For information about the Object Management Group's CORBA 2.6 standard:

<http://www.omg.org>

For information about how to set up the configurations for ORBs other than JacORB, see the documentation for that ORB.

For information about installing this package, see *SRC-PE Getting Started Guide, Chapter 28, Installing the SRC Software on a Solaris Platform*.

Configuring JacORB as the Default ORB

To configure JacORB as the default ORB for the JRE:

1. Access the folder in the SRC software distribution that contains the files you require for the version of JRE that you are using.

- For JRE 1.3, access the folder *SDK/lib-1.3*.

cd /cdrom/cdrom0/SDK/lib-1.3

- For JRE 1.4 or greater, access the folder *SDK/lib-1.4*.

cd /cdrom/cdrom0/SDK/lib-1.4

2. Copy the property files from the folder in the SRC software distribution to the folder *jre/lib* in your JRE installation.

```
cp jacorb.properties <jreInstallDirectory>/jre/lib/jacorb.properties  
cp orb.properties <jreInstallDirectory>/jre/lib/orb.properties
```

3. Copy the appropriate JAR files from the folder in the SRC software distribution to the directory *jre/lib/ext* in your JRE installation.

- For JRE 1.3, copy the file *jacorb.jar*.

```
cp jacob.jar <jreInstallDirectory>/jre/lib/ext/jacob.jar
```

- For JRE 1.4 or greater, copy the file *jacorb.jar*.

```
cp jacob.jar <jreInstallDirectory>/jre/lib/ext/jacob.jar
```

Configuring One Web Application to Use JacORB

To configure a particular Web application that includes the NIC proxy to use JacORB:

1. Access the folder in the SRC software distribution that contains the files you require for the version of JRE that the Web application server is using.

- For JRE 1.3, access the folder *SDK/lib-1.3*.

```
cd /cdrom/cdrom0/SDK/lib-1.3
```

- For JRE 1.4, access the folder *SDK/lib-1.4*.

```
cd /cdrom/cdrom0/SDK/lib-1.4
```

2. Copy the appropriate files from the folder in the SRC software distribution to the folder *WEB-INF/lib* of the Web application.

- For JRE 1.3, copy the files *jacorb.properties* and *jacorb.jar*.

```
cp jacob.properties <webAppDirectory>/WEB-INF/lib/jacob.properties  
cp jacob.jar <webAppDirectory>/WEB-INF/lib/jacob.jar
```

- For JRE 1.4, copy the files *jacorb.properties* and *jacorb.jar*.

```
cp jacob.properties <webAppDirectory>/WEB-INF/lib/jacob.properties  
cp jacob.jar <webAppDirectory>/WEB-INF/lib/jacob.jar
```

3. Configure the NIC factory used by the Web application to use this ORB.

See *Chapter 16, Developing Applications That Use NIC*.

Configuring a Web Application Server to Use JacORB

To configure all Web applications, but not other Java applications, to use JacORB:

1. Access the folder in the SRC software distribution that contains the files you require for the version of JRE that the Web application server is using.
 - For JRE 1.3, access the folder *SDK/lib-1.3*.
cd /cdrom/cdrom0/SDK/lib-1.3
 - For JRE 1.4, access the folder *SDK/lib-1.4*.
cd /cdrom/cdrom0/SDK/lib-1.4
2. For JRE 1.3 and JRE 1.4, include the file *jacorb.jar* file in the classpath for the Web application server.
3. Include the file *jacorb.properties* for the appropriate JRE release in a directory specified in classpath, in the current directory, or in the home directory of the user who starts the Web application server.
4. Configure JacORB to be the ORB for the Web application server ORB. For information about this step, see the JacORB documentation at

<http://www.jacorb.org/documentation.html>

Testing Applications by Using a NIC Proxy Stub

To test an application without NIC, you can configure a NIC proxy stub to take the place of the NIC. The NIC proxy stub comprises a set of explicit mappings of data keys and values in the namespace that contains the NIC proxy properties. When the SRC component passes a specified key to the NIC proxy stub, the NIC proxy stub returns the corresponding value.

For example, you can specify a subscriber's IP address that is associated with a particular SAE. When the SRC component passes this IP address to the NIC proxy stub, the NIC proxy stub returns the corresponding SAE.

Configuring a NIC Proxy Stub from SDX Configuration Editor

For applications that you can configure through SDX Configuration Editor, such as Dynamic Service Activator, you can configure a NIC proxy stub through the Editor.

To use SDX Configuration Editor to enable and configure a NIC proxy stub:

1. In the navigation pane, select a configuration file for the NIC proxy you want to configure.
2. Depending on the configuration file, select the **NIC proxy** tab or the **NIC Proxy** section on another tab.
3. Expand the **NIC Proxy** section, the section for an individual NIC proxy, and then the **Key-Value Pairs When Using NIC Proxy Stub** section.

The screenshot shows the SDX Configuration Editor interface. The 'NIC Proxy (ip)' section is selected and expanded. It contains three sub-sections: 'Resolution', 'Cache', and 'NICHost Selection'. Below these is a 'NicProxyClass' field with a 'Disable' button. A checkbox labeled 'Key-value pairs when using NicProxyStub' is checked. Below the checkbox is a table with 'Property' and 'Value' columns. At the bottom are 'Add', 'Delete', and 'Refresh' buttons.

4. In the NIC Proxy Class field, select **Yes** for NIC Proxy Class, and specify the NIC proxy class for a NIC proxy stub, such as `net.juniper.smgt.gateway.gal.proxy.NicProxyStub`.

See *NIC Proxy Stub Fields* on page 255.

5. Under Key-Value Pairs, enter value(s) that the proxy is to use in the format `key = value` and click **Add**.

See *NIC Proxy Stub Fields* on page 255.

6. Select **File > Save**.
7. Right-click the configuration file, and select **System Configuration > Export to LDAP Directory**.

When you use a NIC proxy stub, you must also configure test data for the stub to use.

See *Configuring the Test Data* on page 256.

NIC Proxy Stub Fields

In SDX Configuration Editor, you can modify the NIC Proxy Class field and the fields in the Key-Value Pairs When Using NIC Proxy Stub section in a NIC Proxy pane or section of a pane in a configuration file.

NIC Proxy Class

- Specifies whether or not the application uses a specified NIC proxy class rather than an actual NIC proxy.
 - Value
 - Enable—Enables a specified NIC proxy class for the application.
 - Disable—Disables a specified NIC proxy class for the application and enables an actual NIC proxy.
 - Guidelines—Enable the specified NIC proxy class. If the NIC proxy class is a value such as `net.juniper.smgmt.gateway.gal.proxy.NicProxyStub`, the NIC proxy stub performs key-value resolutions. Other NIC configuration is not required. A NIC proxy stub is provided for testing purposes.
 - Default—No value
- Property name—`NicProxyClass`

Property

- Type of key for which the NIC proxy stub returns a value.
- Value— < key type >
- Example
 - IP address
 - Subscriber ID

Value

- Value to be returned by the NIC proxy stub for a key specified in the Property field.
- Value— < key value >
- Example—SAE IOR

Configuring a NIC Proxy Stub from SDX Admin

To use SDX Admin to configure a NIC proxy stub:

1. In the navigation pane, select the entry for the NIC proxy.
2. Add the following line to the NIC proxy properties.

Gateway.nic.NicProxyClassName = net.juniper.smgmt.gateway.gal.proxy.NicProxyStub

For example, for Dynamic Service activator, located under *l = DynamicServiceActivation*, *l = WebApplication*, *ou = staticConfiguration*, *ou = Configuration*, *o = Management*, *o = umc*, you would add the lines similar to the following:

```
/nicProxies/ip/Gateway.nic.NicProxyClassName =
net.juniper.smgmt.gateway.gal.proxy.NicProxyStub
/nicProxies/ip/ANY_KEY = corbaloc::192.2.7.100:8801/SAE
```

When you use a NIC proxy stub, you must also configure test data for the stub to use.

See *Configuring the Test Data* on page 256.

Configuring the Test Data

To use a NIC proxy stub, you configure test data for the NIC proxy to use. You can specify that the test data indicate that any key return a specific SAE or that one or more keys map to particular values. If you specify an explicit SAE for a key, the NIC proxy stub returns the IOR for that SAE, rather than the value defined for the ANY_KEY property.

To configure test data, do one of the following:

- Configure a NIC proxy stub to use a corbaloc URL.

See *Configuring a NIC Proxy Stub to Use a corbaloc URL to Test Data* on page 256.

- Configure a NIC proxy stub to use a file URL.

See *Configuring a NIC Proxy Stub to Use a File URL to Test Data* on page 257.

- Configure a NIC proxy stub to use an IOR.

See *Configuring a NIC Proxy Stub to Use an IOR to Test Data* on page 257.

Configuring a NIC Proxy Stub to Use a corbaloc URL to Test Data

To configure a NIC proxy stub to use the corbaloc URL:

1. In the NIC proxy configuration, add a line in the format
corbaloc:: <host> : <port> /SAE

- <host> —Name or IP address of the SAE.
- <portNumber> —TCP/IP port number for the SAE. The default is 8801.

For example, corbaloc::127.0.0.1.145:8801/SAE.

2. In the NIC proxy configuration, add a line to return any key to a specific SAE or a key that the NIC proxy receives.

To return any key, add a line in the format
ANY_KEY = corbaloc:: <host> : <port> /SAE

For example, ANY_KEY = corbaloc::sae1:8801/SAE

To specify explicit mapping between keys and values, add lines in the following format to the NIC proxy configuration.

< mapping > = corbaloc:: < host > : < port > /SAE

For example, the following test data comprises two subscriber IP addresses associated with different SAEs. You define two explicit mappings:

192.0.2.10 = corbaloc::sae1:8801/SAE

192.0.2.11 = corbaloc::sae2:8801/SAE

Configuring a NIC Proxy Stub to Use a File URL to Test Data

To configure a NIC proxy stub to use the IOR file:

1. In the NIC proxy configuration, add a line in the format file:// < absolute path to the IOR file.

For example, file:///opt/UMC/sae/var/run/sae.ior

2. In the NIC proxy configuration, add a line to return any key to a specific SAE or a key that the NIC proxy receives.

To return any key, add a line in the format ANY_KEY = file:// < absolute path to the IOR file > .

For example, ANY_KEY = file:///opt/UMC/sae/var/run/sae.ior

To specify explicit mapping between keys and values, add lines in the following format to the NIC proxy configuration.

< mapping > = file:// < absolute path to the IOR file >

For example, the following test data comprises two subscriber IP addresses associated with the same SAE. You define two explicit mappings:

192.0.2.0 = file:///opt/UMC/sae/var/run/sae.ior

192.0.2.1 = file:///opt/UMC/sae/var/run/sae.ior

Configuring a NIC Proxy Stub to Use an IOR to Test Data

To configure a NIC proxy stub to use a copy of the IOR:

1. Access the *sae.ior* file in the directory */opt/UMC/sae/var/run*.
2. Copy the complete IOR of the SAE from this file.
3. In the NIC proxy configuration, add a line to return any key to a specific SAE or a key that the NIC proxy receives.

To return any key, add a line in the format ANY_KEY = <SAE_IOR> .

- <SAE_IOR> —IOR that you copied

For example:

```
ANY_KEY =
IOR:0000000000000003549444C3A736D67742E6A756E697065722E6E65742
F7361652F5365727669636541637469766174696F6E456E67696E653A312
E30000000000000002000000000000070000102000000000D31302E323
2372E312E323031000022610000001B5374616E64617264496D706C4E616
D652F736165504F412F53414500000000200000000000008000000004
A414300000000010000001C000000000010001000000010501000100010
1090000000105010001000000010000002C000000000000001000000010
000001C00000000000100010000000105010001000101090000000105010
001
```

To specify explicit mapping between keys and values, add lines in the following format to the NIC proxy configuration.

<key> = <value>

For example, the following test data comprises two subscriber IP addresses associated with different SAEs. You can define two explicit mappings:

```
192.0.2.0 =
IOR:0000000000000003549444C3A736D67742E6A756E697065722E6E65742
F7361652F5365727669636541637469766174696F6E456E67696E653A312
E30000000000000002000000000000070000102000000000D31302E323
2372E312E323031000022610000001B5374616E64617264496D706C4E616
D652F736165504F412F53414500000000200000000000008000000004
A414300000000010000001C000000000010001000000010501000100010
1090000000105010001000000010000002C000000000000001000000010
000001C00000000000100010000000105010001000101090000000105010
001
192.0.2.1 =
IOR:0000000000000002438444C3A736D67742E6A756E697065722E6E65742
F7361652F5365727669636541637469766174696F6E456E67696E653A312
E30000000000000002000000000000070000102000000000D31302E323
2372E312E323031000022610000001B5374616E64617264496D706C4E616
D652F736165504F412F53414500000000200000000000008000000004
A414300000000010000001C000000000010001000000010501000100010
1090000000105010001000000010000002C000000000000001000000010
000001C00000000000100010000000105010001000101090000000105010
001
```

Monitoring NIC Proxies

You can use MBeans to monitor NIC proxies. MBeans are a feature of the Java Management Extension (JMX) software. If you want to monitor the MBeans for NIC proxies, your Web application server must include a JMX agent.

NIC proxies create one instance of an MBean called `NicProxyMgmt` to provide information about the role of the NIC proxy to the JMX agent. The way you view the MBeans depends on the particular Web application server and the interfaces that its JMX agent provides. Table 15 shows the information that this MBean provides.

You can reset the values of many `NicProxyMgmt` MBean properties to zero.

To reset the `NicProxyMgmt` MBean properties to zero:

- Execute the reset counters operation through the mechanism that the JMX agent for your Web application server provides.

Table 15 shows which counters the reset operation affects.

Table 15: Information That the `NicProxyMgmt` MBean Provides

Property	Description	Ability to Reset to Zero
<code>nicProxyName</code>	Name of the NIC proxy. Different NIC proxies may exist, providing different functionality.	No
<code>numKeysCachedLocally</code>	Number of key-value pairs that are cached in the NIC proxy (the bigger the cache, the less likely the NIC proxy will have to involve the distributed NIC components in lookups across the network).	No
<code>numLookups</code>	Number of times that the Web application containing this NIC proxy has requested the NIC proxy to look up a data key.	Yes
<code>numLookupErrors</code>	Number of lookups that have failed.	Yes
<code>numKeysNoMatch</code>	Number of lookups in which the provided key does not map to any value.	Yes
<code>numKeysOneMatch</code>	Number of lookups in which the provided key maps to exactly one value.	Yes
<code>numKeysMultiMatch</code>	Number of lookups in which the provided key maps to more than one value.	Yes
<code>lookupTimeAvg</code>	For the 100 most recent (successful and unsuccessful) lookups, the average time (in milliseconds) of the lookup.	Yes
<code>lookupTimeMin</code>	For the 100 most recent (successful and unsuccessful) lookups, the minimum time (in milliseconds) of the lookup.	Yes
<code>lookupTimeMax</code>	For the 100 most recent (successful and unsuccessful) lookups, the maximum time (in milliseconds) of the lookup.	Yes

Chapter 16

Developing Applications That Use NIC

This chapter describes how to develop an external application to interact with a network information collector (NIC). Topics include:

- External Application Requirements for NIC on page 261
- External Non-Java Applications That Use NIC on page 261
- External Java Applications That Use NIC on page 263
- Updating Information About Address Pools on page 269

External Application Requirements for NIC

If you write an external application to use NIC to perform a resolution, you can include NIC functionality in one of the following ways:

- For non-Java applications, use the interface module `NicAccess`, an IDL file that provides access to the NIC locator feature. The NIC locator can resolve the value of one or more keys.
- For Java applications, include the NIC proxy client libraries to use NIC in client/server mode.
- For Java applications, include the NIC proxy client libraries and the NIC host client libraries to use NIC in local host mode.

External Non-Java Applications That Use NIC

If you write an application in a language other than Java, you can use the NIC access interface module, a simplified CORBA interface, to perform one or more resolutions. By using this interface you can access through CORBA NIC locators, NIC proxies that run within the NIC host. The configuration properties for NIC locators are similar to those for NIC proxies in applications such as aggregate services and the sample residential portal.

For information about the NIC access interface module, see the API documentation in the SRC software distribution in the folder *SDK/doc/idl/nic* or on the Juniper Networks Web site at

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

Creating a NIC Locator to Include with a Non-Java Application

A NIC locator provides the same functionality as a NIC proxy, but is designed to work with non-Java applications.

You use the NIC access interface module to include NIC locators with your application by compiling the IDL file with your application files.

To use the NIC access interface module to create NIC locators:

1. Connect to the directory.
2. Obtain a CORBA reference to the NIC access interface from one of the following:

- The access IOR provided in the directory in the dynamic configuration DN under the hostname—typically, *host/demohost*.

You can read this information from SDX Admin from a host under *ou = dynamicConfiguration, ou = Configuration, o = Management, o = umc*.

- A corbaloc URL in the format:

```
corbaloc::<host>:8810/Access
```

3. From the NIC access interface module, obtain a NIC locator, as identified by *NicFeature*. For example:

```
feature = access.getLocatorFeature(nicNameSpace); //nicNameSpace example
"/nicLocators/ip"
```

In the NIC configuration scenarios, the syntax for a NIC locator is */nicLocators/<NIC key type>* where.

- **nicLocators**— Specifies all of the NIC locators in a NIC host.
- **<NIC key type>**— Specifies the type of data that the key provides for the NIC resolution, such as ip, login, DN.

To view information about the NIC locators included in a NIC scenario, see *Chapter 20, Reviewing the NIC Configuration*.

4. Search for the key. For example:

```
feature.lookupSingle(NicLocatorKey key) //NicLocatorKey is coming from the IDL
```

For information about the NIC access interface module, see the API documentation in the SRC software distribution in the folder *SDK/doc/idl/nic* or on the Juniper Networks Web site at

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

External Java Applications That Use NIC

If you write an external Java application that interacts with a NIC, include NIC libraries in the application. These libraries are for NIC proxies and local NIC hosts. These libraries are located in the SRC distribution under *SDK/lib/nic*.

Typically, each NIC resolution process requires one NIC proxy. For example, the OnePopLogin sample data includes two resolution processes:

- Mapping of a subscriber's IP address to the subscriber's login name
- Mapping of the subscriber's login name to the SAE reference

An application that uses both these resolution processes would require two NIC proxies.

The NIC proxy provides a simple Java interface, the NIC application programming interface (API). You configure the NIC proxy to communicate with one resolver. For efficiency if you use NIC in client/server mode, the NIC proxy caches the results of resolution requests so it can respond to future requests for the same key without contacting the resolver.

The SRC software includes a factory interface, the NIC factory, to allow applications to instantiate, access, and remove NIC proxies. It also includes JAR files for NIC client and NIC host libraries.

Developing a Java Application to Communicate with a NIC Proxy

You must configure an application to communicate with a NIC proxy.

If you are using Java Runtime Environment (JRE) 1.3 or higher, you must include in your application the Java archive (JAR) files, which are in the SRC software distribution in the folder */SDK/lib/* with your application:

Configuration tasks that use the API calls to communicate with the NIC proxy are:

1. Instantiating a Configuration Manager on page 264
2. Passing a Reference to the Configuration Manager to the NIC Factory on page 264
3. Instantiating the NIC Factory Class on page 264
4. (Optional) Initializing Logging on page 265
5. Instantiating the NIC Proxy on page 266
6. Managing a Resolution Request on page 266
7. Deleting Invalid Results from the NIC Proxy's Cache on page 268
8. Removing the NIC Proxies on page 268

For more information about the API calls, see the online documentation in the SRC software distribution in the folder */SDK/doc/nic* or on the Juniper Networks Web site at

<http://www.juniper.net/techpubs/software/management/sdx/api-index.html>

Instantiating a Configuration Manager

The application must instantiate a configuration manager.

To enable the application to instantiate a configuration manager to obtain a NIC instance from the NIC factory:

- Call one of the following methods:
 - For some applications (other than Web applications), in which you must define the system property `-DConfig.bootstrapFilename`, you can call the following method:

```
ConfigMgr configMgr = ConfigMgrFactory.getConfigMgr();
```

- For Web applications, you can instantiate the configuration manager as follows:

```
ConfigMgr configMgr = ConfigMgrFactory.getConfigMgr(properties);
```

- `properties`—`java.util.Properties` object, typically the bootstrap file, which contains all the configuration properties for the NIC proxy.

Passing a Reference to the Configuration Manager to the NIC Factory

To pass a reference to the configuration manager to the NIC factory class:

- Call the following method in the application:

```
NicFactory.setConfigManager(configMgr);
```

Instantiating the NIC Factory Class

The way you instantiate the NIC factory depends on the object request broker (ORB) configuration:

- If the NIC proxy uses the default ORB, call the following method in the application:

```
NicFactory nicFactory = NicFactory.getInstance();
```

This code instantiates a new NIC factory. Unless the `NicFactory.destroy` method has been called, subsequent calls to this method will return the instantiated NIC factory.

- If the NIC proxy does not use the default ORB, call the following method:

```
NicFactory.initialize(props);
NicFactory nicFactory = NicFactory.getInstance();
```

- props—java.util.Properties object, which contains the ORB properties for the NIC proxy. For example, if the NIC proxy uses JacORB but JacORB is not the default ORB, the ORB properties are:

```
org.omg.CORBA.ORBClass=org.jacorb.orb.ORB
org.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
```

This code will instantiate a new NIC factory using the specified ORB. Unless the application has called the `NicFactory.destroy` method, subsequent calls to the `getInstance()` method will return the instantiated NIC factory. However, if the application has called the `destroy()` method, it must recall the `initialize()` method before it can call the `getInstance()` method.

For information about the `NicFactory.destroy` method, see *Removing the NIC Proxies* on page 268.

Initializing Logging

You must initialize logging only if you want to view the logging information produced by the NIC proxy.

To enable the application to initialize logging:

- Call the following method:

```
Log.init(configMgr, configNameSpace);
```

- configMgr—Instance of the configuration manager, the value returned from the `getConfigMgr()` method
- configNameSpace—String that specifies the configuration namespace where you defined the logging properties

- If you define the logging properties in the bootstrap file, specify the root namespace, `"/`.

```
Log.init(configMgr, "/");
```

- If you define the logging properties in the directory, specify the namespace relative to the property `Config.net.juniper.smgmt.lib.config.staticConfigDN`, which you configure in the bootstrap file.

```
Log.init(configMgr, "/Applications/Quota");
```

Instantiating the NIC Proxy

To enable the application to instantiate a NIC proxy:

- Call the following method:

```
NIC nicProxy = nicFactory.getNicComponent(nicNameSpace, configMgr)
```

Alternatively, if the expected data value (specified for the property `nic.value` in the NIC proxy configuration) is an SAE reference, you can call the following method:

```
SaeLocator nicProxy = nicFactory.getSaeLocator(nicNameSpace, configMgr);
```

- `nicFactory`—Instance of the NIC factory
- `nicNameSpace`—String that specifies the configuration namespace where you defined the properties for the NIC proxy

- If you define the NIC properties in the bootstrap file, specify the root namespace, `"/`.

```
NIC nicProxy = nicFactory.getNicComponent("/", configMgr)
```

- If you define the properties in the directory, specify the namespace relative to the property `Config.net.juniper.smgmt.lib.config.staticConfigDN`, which you specified in the bootstrap file.

```
NIC nicProxy = nicFactory.getNicComponent("/Applications/Quota", configMgr)
```

- `configMgr`—Instance of the configuration manager, the value returned from the `getConfigMgr()` method

Managing a Resolution Request

To enable the application to submit a resolution request and obtain the associated values:

1. Construct a `NicKey` object to enable the application to pass the data key to the NIC proxy:

```
NicKey nicKey = new NicKey(stringKey);
```

- `stringKey`—Data key for which you want to find corresponding values.

For the syntax of allowed data types, see *Chapter 18, NIC Resolution Process*.

2. If the resolution process specifies constraints that you wish to provide in the resolution request, add them to the `NicKey` object:

```
NicKey.addConstraint(constName, constValue);
```

- `constName`—Name of the constraint.

For the allowed data types and their syntax, see *Chapter 18, NIC Resolution Process*.

- `constValue`—Specific value of the constraint.

For the allowed syntax for the data types, see *Chapter 18, NIC Resolution Process*.

3. Call a method that starts the resolution process.

For example, you can call a method specified in the NIC interface:

```
NicValue val = nicProxy.lookupSingle(nicKey);
```

Alternatively, if the expected data value is an SAE reference, you can call the following method:

```
Saeld saeld = nicProxy.lookupSae(nicKey);
```

4. Call the `getValue` method to access the string representation of the data value obtained by the NIC proxy.

```
String val=val.getValue();
```

Alternatively, if the expected data value is an SAE reference:

```
String val=saeld.getValue();
```

5. (Optional) Call a method to get intermediate values obtained during a resolution.

- Call the `getIntermediateValue` method if the application expects only one value. This method takes the name of a data type and returns as a string the first value it finds.

```
String getIntermediateValue(String dataTypeName){};
```

For information about data types, see *Chapter 18, NIC Resolution Process*.

- Call the `getIntermediateValues` or `getAllIntermediateValues` method if the application expects multiple values. These methods take the name of a data type and return values as follows:
 - The `getIntermediateValues` method returns a list of values as a string array.

```
String[] getIntermediateValues(String dataTypeName){};
```

For information about data types, see *Chapter 18, NIC Resolution Process*.

- The `getAllIntermediateValues` method returns a map of all intermediate values for the request. The key for the map is the name of the network data type, and the value of the map is a string array of the intermediate values.

```
Map getAllIntermediateValues();
```

Deleting Invalid Results from the NIC Proxy's Cache

If the application receives an exception when using values that the NIC proxy returned for a specific key, it must inform the NIC proxy to delete this entry from its cache.

To enable the application to inform the NIC proxy to delete an entry from its cache:

- Call the following method:

```
nicProxy.invalidateLookup(nicKey, nicValue);
```

- `nicKey`—Data key that you want to remove from the cache
- `nicValue`—Optional data value that corresponds to this key

If the application passes a null data value to the NIC proxy, the NIC proxy removes all the values associated with the data key from its cache.

Removing the NIC Proxies

Make sure that before your application shuts down, it removes the NIC proxy instances to release resources for other software processes.

To remove one NIC proxy instance:

- Call the following method:

```
NicProxy.destroy();
```

To remove all NIC proxy instances, call the following method:

```
NicFactory.destroy();
```

Updating Information About Address Pools

If you associate an existing address pool with an interface and you do not want to wait for this new information to be propagated based on the Cache Entry Age property of the NIC proxy or the Event Life Expectancy property of the agents, then you must manually clear the NIC proxy cache.

To clear the NIC proxy cache when an application is deployed in a J2EE container that supports Java Management Extension (JMX) software, do one of the following:

- Use the NicProxyMgmt MBean.
- Restart the application.
- Restart the application server.

For information about modifying the NIC proxy cache properties, see *Chapter 13, Configuring Applications to Communicate with an SAE*.

For information about modifying the event life expectancy for agents, see *Chapter 20, Reviewing the NIC Configuration*.

Chapter 17

Configuring NIC Host Redundancy

Typically, you use network information collector (NIC) replication rather than NIC host redundancy to maintain high availability for NIC. NIC replication lets you share system load between NIC hosts and is simpler to configure and maintain than NIC redundancy. NIC redundancy is provided for backward compatibility. For information about NIC replication, see *Chapter 9, Locating Subscriber Information with the NIC*.

Topics include:

- Overview of NIC Host Redundancy on page 271
- Before You Configure NIC Host Redundancy on page 272
- Configuring NIC Host Redundancy on page 272
- Configuring Monitors on page 275
- Optimizing Performance of the NIC Proxy for NIC Host Redundancy on page 280
- Viewing Log Files for NIC Hosts and Monitors from the Solaris CLI on page 282
- Clearing Persistent Data and Logs for NIC Hosts and NIC Monitors on page 282

Overview of NIC Host Redundancy

NIC host redundancy is available only on installations on Solaris platforms. You configure NIC host redundancy from the SDX Configuration Editor and from the local configuration tool for NIC monitors. You cannot configure NIC host redundancy from the CLI.

In NIC host redundancy, two NIC hosts with the same configuration act as a redundant pair. One host assumes the active role, and the other the passive role. If the active host fails, the passive host writes its Common Object Request Broker Architecture (CORBA) object reference to the directory and assumes the active role. The former active host reassumes the active role only when the current active host fails. Each pair of redundant hosts constitutes a *community*.

You can also add a *monitor*, which tracks the redundant components, to that community. The SRC software includes a NIC monitor component; you can install one NIC monitor on a machine. You can then add that monitor to the directory and use it in a host redundancy configuration.

The monitor acts on behalf of a client or server that is not part of the community. If the client or server cannot communicate with the active host in the community but the hosts can communicate with each other, the monitor prompts the passive host to assume control. Without the monitor, the passive host would not assume control, and the host would be unavailable.

The members in the community exchange keepalive messages to monitor each other's availability. If a member does not receive a keepalive message from another member during the specified time, the first member assumes that the second member is unavailable.

For an example of NIC host redundancy, see the network configuration for the OnoPopAllRealms distributed configuration shown in *Chapter 21, NIC Configuration Scenarios*. The monitor BOClient is installed on the same machine as the NIC proxy and tracks the communications between the hosts OnePopBO/One and OnePopBO/Two. If the monitor detects that the connection to the active host is unavailable, it prompts the passive host to assume control.

Before You Configure NIC Host Redundancy

If a directory does not support multimaster replication on the same machine as the NIC host, use NIC replication rather than NIC host redundancy. Without multimaster replication on the same machine, the NIC host must write its CORBA object references to the directory only when you start the NIC host, therefore a directory failure is unlikely to affect the operation of the NIC.

Before you configure NIC host redundancy, make sure that:

- NIC host software is installed on two separate machines for each set of NIC hosts to act as a redundant pair.
- Redundancy monitor software is installed on each system that is to act as a NIC monitor.

Usually, you install the monitor on the same machine as the SAE. A machine can support only one monitor.

Configuring NIC Host Redundancy

To configure NIC host redundancy, complete the following tasks:

1. Configuring Redundant Hosts on page 273
2. Configuring Communities on page 274

Configuring Redundant Hosts

If you use NIC host redundancy, configure two redundant hosts for each host.

To use SDX Configuration Editor to configure redundant hosts:

1. In the navigation pane, select a configuration file for the NIC.
2. Select the **Hosts** tab, expand a section for a host, and then expand the **Redundant Hosts** section.
3. Create a new instance of a redundant host.

The screenshot shows a configuration window titled "Redundant Hosts". At the top, there are two buttons: "Create a New Instance of Redundant Host" and "Delete an Instance". Below these buttons is a section titled "Redundant Host (one)". This section contains three input fields: "Redundancy Community", "Hot Standby Agents", and "Hosted Agents".

4. In the Redundant Host section, edit the fields.
See *Redundant Hosts Fields* on page 273.
5. Repeat Steps 2 to 4 to create the other redundant hosts.
6. Select **File > Save**.
7. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Redundant Hosts Fields

In SDX Configuration Editor, you can modify the following fields in the Redundant Hosts section of the Hosts pane in a NIC configuration file.

Redundancy Community

- Path to the community to which the redundant host belongs.
- Value—Path to community
 - Path is relative to the Static Configuration object.
 - Subfolders in a path are separated by the forward slash (/).
- Example—/communities/nicsae

Hot Standby Agents

- List of paths to redundant agents for SAE plug-in agents supported by the redundant hosts.
- Value—Comma-separated list of paths to agents
 - Path is relative to the Static Configuration object.
 - Subfolders in a path are separated by the forward slash (/).
- Example—agents/IpVr/demo, agents/DnVr/demo

Hosted Agents

- List of paths to agents and redundant agents supported by the redundant hosts.
- Value—Comma-separated list of paths to agents
 - Path is relative to the Static Configuration object.
 - Subfolders in a path are separated by the forward slash (/).
- Example—agents/IpVr, agents/IpVr/demo, agents/DnVr, agents/DnVr/demo, agents/PoolVr

Configuring Communities

To use SDX Configuration Editor to configure a community for NIC host redundancy:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Redundancy** tab, expand the **Communities** section, and create a new instance of a community.

The screenshot shows a software interface with a 'Communities' section. Under this section, there is a 'Community (nicsae)' entry. Above this entry are two buttons: 'Create a New Instance of' and 'Delete an Instance', each with a checkmark icon. Below the 'Community (nicsae)' entry, there are two input fields: 'Community Members' and 'Keep Alive Time'.

3. In the Communities section, edit the values for the fields.
See *Communities Fields* on page 275.
4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Communities Fields

In SDX Configuration Editor, you can modify the following fields in the Communities section of the Redundancy pane in a NIC configuration file.

Community Members

- List of community members—either two redundant hosts or two redundant agents plus an optional monitor.
- Value—Text string
 - For hosts—Name of the redundant host in the format `<primaryHostName> / <redundantHostName>`
 - For agents—Name of the redundant host that supports the redundant agent in the format `<primaryHostName> / <redundantHostName>`
 - For monitor—Name of the monitor; must match the name specified in the monitor process (see *Configuring a Monitor Process* on page 277)
- Example—`DemoHost/One, DemoHost/Two, DemoMonitor`

Keep Alive Time

- Maximum time after which a host or agent in the community must receive an availability message from the other host or agent. If the secondary host or agent does not receive a message during this time, it assumes that the primary host or agent is unavailable and assumes control.
- Value—Number of seconds in the range 0–2147483647
- Example—`agents/IpVr/demo, agents/DnVr/demo`

By design, redundancy properties are not dynamically updated if their value changes in the directory. If you change the configuration for the redundancy communities while the two hosts are running but before you start the monitor, the configuration perceived by the monitor is different from the one perceived by the hosts.

Configuring Monitors

You can add NIC monitors to track redundant components.

Complete the following tasks to configure NIC monitors:

1. Configuring Operating Parameters for Monitors on page 276
2. Configuring a Monitor Process on page 277
3. Configuring JacORB Properties on Redundant NIC Hosts on page 278
4. Starting NIC Hosts on page 279
5. Starting NIC Monitors on page 279

Configuring Operating Parameters for Monitors

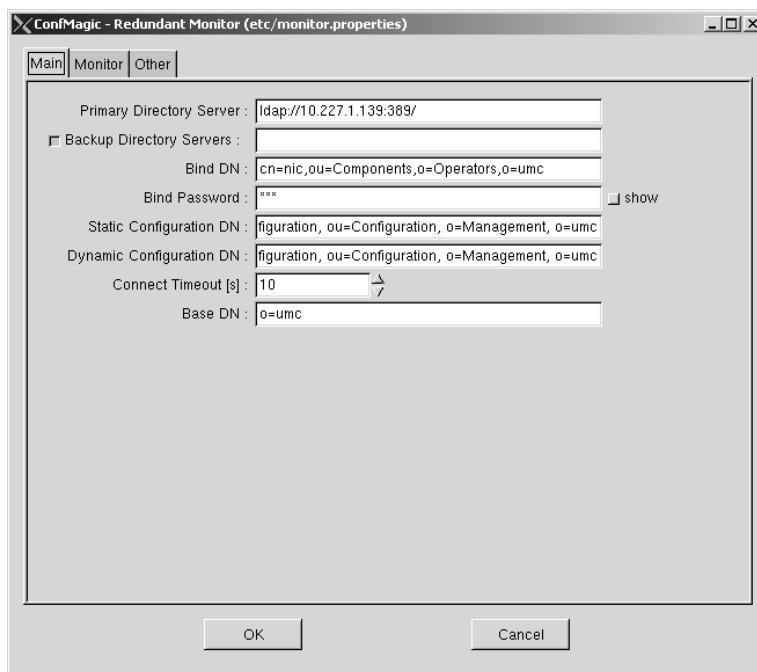
You must configure operating parameters for each monitor in your NIC configuration. The operating parameters define how the monitor interacts with other SRC components, such as the directory.

To configure the operating parameters:

1. Log in as root.
2. Start the local configuration tool in the directory where you installed the redundancy monitor software.

/opt/UMC/monitor/etc/config

The Redundant Monitor window appears.



3. In the Redundant Monitor window, edit or accept the default values for the fields.

See *JRE Properties for a Redundant Monitor* on page 277 and *Chapter 11, Configuring NIC on a Solaris Platform*.

4. Click **OK**.

JRE Properties for a Redundant Monitor

In the local configuration tool, you can set the value for the Redundant Monitor Java in the Monitor tab.

Redundant Monitor Java

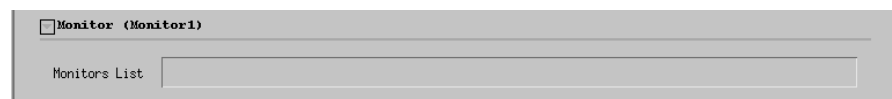
- Path to the JRE.
- Value—Path (absolute or relative) to the directory that contains the JRE
- Example—`../jre/bin`

Configuring a Monitor Process

You configure a monitor by adding it to a monitoring process, referred to as a monitor.

To use SDX Configuration Editor to configure a monitoring process:

1. In the navigation pane, select a NIC configuration file.
2. In the Redundancy tab, expand the **Monitors** section, and create a new instance of a monitor (monitoring process).



3. In the Monitor section, specify the monitors that this monitoring process supports.

See *Monitor Field* on page 278.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Monitor Field

In SDX Configuration Editor, you can modify the following field in the Monitor section of the Redundancy pane in a NIC configuration file.

Monitor List

- List of monitors that the monitoring process will support.
- Value—Comma-separated list of monitors; each monitor has the format `< monitorName > :/communities/ < communityName >`
 - `< monitorName >` —Name of monitor; must match the name specified in the community (see *Configuring Communities* on page 274)
 - `< communityName >` —Name of community; must match the name specified in the configuration for the community (see *Configuring Communities* on page 274)
- Example—`DemoMonitor:/communities/Demohost, nicSaeDemoMonitor:/communities/nicsaeDemo`

Configuring JacORB Properties on Redundant NIC Hosts

If the system on which a NIC hosts resides uses an object request broker (ORB) different from the one provided in the SRC software distribution, it is important to configure some JacORB properties on each NIC host to enable the ORB to correctly determine when connections between redundant NIC hosts are unavailable. If you do not configure these properties properly, the ORB relies on the TCP default socket timeout (usually 8 minutes) to determine when a connection between redundant NIC hosts is unavailable or cannot be established. Using high values for these properties affects overall system availability.

To configure the JacORB properties:

1. With a text editor, open the *jacorb.properties* file.
2. Ensure that the property `jacorb.connection.client_idle_timeout` is uncommented, and set this property to a value between 5,000 and 30,000 milliseconds.

See *JacORB Properties for Timeouts* on page 279.

3. (JRE 1.4 only) If the default value is not appropriate for your network configuration, change the value of the property `net.juniper.smgmt.socket_factory.timeout.connect`.

See *JacORB Properties for Timeouts* on page 279.

4. Save and close the *jacorb.properties* file.
5. Repeat Steps 1 to 4 for each NIC host.

JacORB Properties for Timeouts

In the *jacorb.properties* file, you can modify the following timeout properties.

jacorb.connection.client_idle_timeout

- Time after which the ORB assumes that an existing connection between two machines has become unavailable.
- Value—Number of milliseconds in the range 5000–30000
- Default—30000
- Example—`jacorb.connection.client_idle_timeout = 10000`

net.juniper.smgmt.socket_factory.timeout.connect

- Time after which the ORB assumes that a new connection cannot be established between two machines.
- Value—Number of milliseconds in the range 0–2147483647
- Default—5000
- Example—`net.juniper.smgmt.socket_factory.timeout.connect = 10000`

Starting NIC Monitors

You must start each NIC monitor before its associated components can begin operating.

To start a monitor:

1. On the machine on which the monitor is installed, log in as `root` or as an authorized nonroot admin user.
2. Start the monitor from its installation directory:

`/opt/UMC/monitor/etc/monitor start`

After you have started a monitor, you can view log files of its operation and stop the host or monitor.

Starting NIC Hosts

You must configure operating parameters for each NIC host and then start each NIC host before its associated components can begin operating.

For information about how to configure NIC host parameters and how to start NIC hosts see *Chapter 11, Configuring NIC on a Solaris Platform*.

Verifying That a Monitor Is Running

To verify that the monitor is running:

1. On the host on which the monitor is installed, log in as **root** or as an authorized nonroot admin user.
2. Display the status of the monitor from its installation directory.

```
/opt/UMC/monitor/etc/monitor status
```

Stopping a Monitor

To stop the monitor:

1. On the host on which the monitor is installed, log in as **root** or as an authorized nonroot admin user.
2. Stop the monitor from its installation directory.

```
/opt/UMC/monitor/etc/monitor stop
```

Optimizing Performance of the NIC Proxy for NIC Host Redundancy

If you configure NIC host redundancy and the NIC proxy submits resolution requests to the NIC host at a high rate, the NIC proxy may take a significant amount of time to start communicating with a NIC host when that host makes the transition from the passive role to the active role. In this case, you will notice that many resolution requests are failing, and the NIC log files will indicate that the NIC proxy is sending requests to the former active NIC host.

This time delay occurs because the ORB tries to connect the NIC proxy to the former active NIC host for each outstanding resolution request. When the connection attempt fails, the ORB tries several times to reconnect the devices and waits for a specified interval of time between each attempt. To accelerate this process, modify the ORB properties that control these actions.

Depending on which NIC proxies are affected, you can modify the properties for one Web application only or for all Web applications in a JRE. For information about configuring the properties of an ORB other than JacORB, see the documentation for the ORB, and use the property descriptions below as a guideline.

To configure the properties for JacORB:

1. Access the directory that contains the *jacorb.properties* file.
 - To configure the properties for one Web application only:

cd <webAppDirectory>/WEB-INF/lib
 - To configure the properties for all Web applications in the JRE:

cd <jreInstallDirectory>/jre/lib
2. With a text editor, open the *jacorb.properties* file.
3. Modify the values of the properties *jacorb.retries* and *jacorb.retry_interval*.

See *JacORB Properties for Retry Intervals* on page 281.
4. Save and close the *jacorb.properties* file.

JacORB Properties for Retry Intervals

In the *jacorb.properties* file, you can modify the following properties for retry intervals.

jacorb.retries

- Number of times that the ORB tries to connect the NIC proxy to the NIC host.
- Value—Integer in the range 0–2147483647
- Guidelines—If you notice that many resolution requests are failing and the NIC log files indicate that the NIC proxy is sending requests to the former active NIC host, reduce the value of this property.
- Default—5
- Example—*jacorb.retries* = 1

jacorb.retry_interval

- Time between the ORB's attempts to connect the NIC proxy to the NIC host.
- Value—Number of milliseconds in the range 0–2147483647
- Guidelines—If you notice that many resolution requests are failing and the NIC log files indicate that the NIC proxy is sending requests to the former active NIC host, reduce the value of this property.
- Default—500
- Example—*jacorb.retry_interval* = 200

Viewing Log Files for NIC Hosts and Monitors from the Solaris CLI

To view the log files for the UNIX processes associated with NIC hosts and monitors:

1. Access the log directory.

```
cd /opt/UMC/nic/var/log
```

or

```
cd /opt/UMC/monitor/var/log
```

2. Display the file.

```
more nic_info.log
```

or

```
more moninfo.log
```

Clearing Persistent Data and Logs for NIC Hosts and NIC Monitors

You can clear the log files for a NIC host or monitor and delete the persistent data that the NIC host or monitor writes to files or devices by using the **stdout** and **stderr** options. Clearing these types of data does not remove data about NIC hosts or monitors from the directory.

To delete persistent data and clear log files for a NIC host or monitor:

1. On the machine where you installed the NIC host or monitor, log in as **root** or as another authorized user.
2. Stop the NIC host or monitor.

```
/opt/UMC/nic/etc/nichost stop  
/opt/UMC/monitor/etc/monitor stop
```

3. Delete the data.

```
/opt/UMC/nic/etc/nichost clean  
/opt/UMC/monitor/etc/monitor clean
```

The system responds with a status message.

4. Restart the NIC host.

```
/opt/UMC/nic/etc/nichost start  
/opt/UMC/monitor/etc/monitor start
```


Chapter 18

NIC Resolution Process

This chapter provides information about the NIC resolution process. You should be familiar with this information if you customize a NIC configuration scenario. Topics include:

- Overview of the Resolution Process on page 283
- NIC Data Types on page 285
- Constraints as NIC Data Types on page 287

Overview of the Resolution Process

Because NIC can process all types of network data, you must use different resolution processes for different types of data mappings to maximize the performance of the NIC configuration. Resolving data requests consumes significant resources.

Table 16 shows the resolutions that the components in the NIC configuration scenarios perform. For customized types of resolutions, contact Juniper Networks Professional Services.

Table 16: Available NIC Resolutions

Key	Value
Subscriber's IP address (JUNOS routing platform)	SAE reference
Subscriber's IP address	Subscriber's login name
Subscriber's IP address	SAE reference
Subscriber's login name	SAE reference
Subscriber's username	SAE reference
Access DN	SAE reference

NIC Realms

Each resolution process and the resolvers that perform that process are defined by a *realm*—a group of resolvers that perform a series of resolution tasks to provide a mapping from a specified key to a specified data type. For example, the sample data provided for the NIC includes a realm called *dn* in which the resolution process takes an access subscriber's distinguished name (DN) as the key and returns a reference to the SAE managing this subscriber as the value.

A set of hosts in a NIC can support multiple realms. Similarly, the agents in a NIC can support more than one realm. However, you can assign a resolver only to one realm.

A NIC host can support NIC resolvers for multiple realms. Consequently, you can simplify the NIC configuration and minimize the use of network resources by limiting the number of NIC hosts in your NIC configuration. NIC hosts can also handle multiple NIC resolvers in the same realm. In this case, when a NIC host receives a request, it chooses a NIC resolver as follows:

1. It identifies the NIC resolvers that are available to process the request.
2. If multiple NIC resolvers are available, it obtains a cost value associated with the resolution process from each resolver and selects the resolver that has the lowest cost value.

Key to Value Resolution

A resolution process typically defines several transitions or *roles*, with each transition resolving a NIC key to a value. For example, the resolution process to identify the SAE that manages a particular subscriber based on that subscriber's IP address involves the following roles:

1. Given the IP address, determine the IP address pool.
2. From the IP address pool, determine the VR.
3. From the VR, determine the SAE that manages that VR.

A role specifies the types of data with which it works. NIC supports a number of data types, including one that lets you add an identifier to other data types to let you specify different values for one data type.

For information about NIC data types, see *NIC Data Types* on page 285 and *Constraints as NIC Data Types* on page 287.

NIC Data Types

The NIC supports the data types that appear in the following list. You can qualify these data types by adding an identifier to:

- Distinguish between different instances of a data type in a resolution scenario.
- Provide information about a data type to clarify the use of that data type in a resolution.

For information about qualifying data types, see *Chapter 19, Customizing a NIC Configuration*.

AnyString

- Generic data type to represent the information that you want to collect.
- Value—Alphanumeric characters
- Guidelines—You can qualify this data type with an identifier to provide information about the type of data that AnyString represents.
- Example—My(IP), My(Vr)

Dn

- DN of an access.
- Value—DN
- Example—*accessName = PrimaryAccess, enterpriseName = juniper, ou = Sunnyvale, retailerName = VPNprovider, o = Users, o = umc*

Domain

- Domain name.
- Value—Name of a domain
- Example—Example.net

Enterprise

- DN of an enterprise.
- Value—DN
- Example—*enterpriseName = juniper, ou = Sunnyvale, retailerName = VPNprovider, o = Users, o = umc*

Router

- Name of router.
- Value—Text string
- Example—router1

Interface

- Name of a router's interface.
- Value— < interfaceName > @ < vrName > @ < routerName >
 - < interfaceName > —Name of the interface, such as fastEthernet 3/1 , exactly as it is configured on the router
 - < vrName > —Name of VR exactly as it is configured on the router
 - < routerName > —Name of router exactly as it is configured on the router
- Example—FastEthernet4/1.0@boston@router1

InterfaceId

- Identifier of an interface.
- Value— < intfIndex > @ < routerName >
 - < intfIndex > —Simple Network Management Protocol (SNMP) index of the interface
 - < routerName > —Name of router exactly as it is configured on the router
- Example—4@router1

Ip

- Subscriber's IP address.
- Value—IP address
- Example—192.0.2.10

IpPool

- IP address pool.
- Value—Range of IP addresses enclosed in square brackets and parentheses
- Guidelines—If you enter an IP address that includes a value greater than 255 in one octet of the address, that part of the address is masked to fit the eight bits.
- Example—([192.0.2.0 192.0.2.255])

Saeld

- SAE reference.
- Value—CORBA interoperable object reference (IOR) for SAE
- Example—IOR:0000000000000002438444C3A736...

Vr

- Name of the virtual router.
- Value— < vrName > @ < routerName >
 - < vrName > —Name of VR exactly as it is configured on the router
 - < routerName > —Name of router exactly as it is configured on the router
- Example—vr1 @router1

Constraints as NIC Data Types

Constraints are data types that a resolver uses when it executes a role. You can define:

- Multiple constraints for a role—Software performs an OR operation to determine whether the constraint is met.
- Multiple data types in a constraint—Software performs an AND operation to determine whether the multiple constraints are met.

Constraints can be either mandatory or optional. If a constraint is mandatory and the resolver for the role does not receive an appropriate value in the data request, the resolver must obtain the constraint value from other NIC resolvers. However, if a constraint is optional and the resolver for the role does not receive an appropriate value in the data request, the resolver can execute its role without the constraint value. In this case, the resolver may obtain multiple values for the data key, and the NIC host responds to the NIC proxy as follows:

- If the request is for multiple results, the host provides all the results.
- If the request is for one result and the resolution process returns different results, the host returns an error message.
- If the resolution process returns multiple instances of the same result, the resolver provides only one result.

For example, if you want to obtain an SAE reference for a subscriber's IP address, you could define the following roles:

1. From the IP address, determine the VR (mandatory constraint IpPool).
2. From the VR, determine the SAE that manages that VR.

Because the first step has a mandatory constraint, the resolver for this role must use the IP pool supplied in the request, or obtain the IP pool from another resolver that determines IP pools from IP addresses. So you must define an extra step at the start of the resolution process:

1. From the IP address, determine the IP pool.
2. From the IP address, determine the VR (mandatory constraint IpPool).
3. From the VR, determine the SAE that manages that VR.

Chapter 19

Customizing a NIC Configuration

In most cases, you use the network information collector (NIC) configuration scenarios supplied with the SRC software. If you need a different resolution scenario, you can customize these scenarios for a NIC that runs on a Solaris platform. Topics include:

- Before You Customize a NIC Configuration on page 289
- Planning a Custom NIC Configuration on page 290
- Creating a Custom NIC Configuration by Adding Components to an Existing Scenario on page 290
- Creating a Custom NIC Configuration by Removing Components in an Existing Scenario on page 292
- Qualifying NIC Data Types on page 293
- Managing Directory Changes for the Directory Agent on page 295

Before You Customize a NIC Configuration

We recommend that you customize a NIC configuration by adding components to or removing components from a NIC configuration scenario that is supplied with the sample data.

Before you customize a NIC configuration:

- Review the configuration scenarios provided with the SRC software to determine whether you must use a NIC configuration scenario different from the ones supplied.

See Chapter 9, Locating Subscriber Information with the NIC.

- Make sure that you are familiar with the NIC resolution process.

See Chapter 18, NIC Resolution Process.

Planning a Custom NIC Configuration

Typically, you combine components from existing NIC configuration scenarios to create a new one.

To plan how to combine NIC configuration scenarios:

1. Review the NIC configuration scenarios available in the sample data, and decide which scenarios provide resolutions that you want to use. See *Chapter 9, Locating Subscriber Information with the NIC*.
2. In SDX Admin, review the configuration for each scenario to be used. The configuration scenarios appear under *l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc*.

If both scenarios use agents that have the same name, review the agent configuration to see whether or not the configurations are the same.

3. Decide whether a non-Java application is to communicate with NIC host.
4. Make a list of which agents and resolvers (stored under *l = realms*) to use from each configuration scenario.
5. Decide which configuration scenario to use as a base. You will make a copy of this scenario and modify it.

Creating a Custom NIC Configuration by Adding Components to an Existing Scenario

You can create a custom NIC configuration scenario in SDX Admin by copying an existing configuration scenario and modifying it.

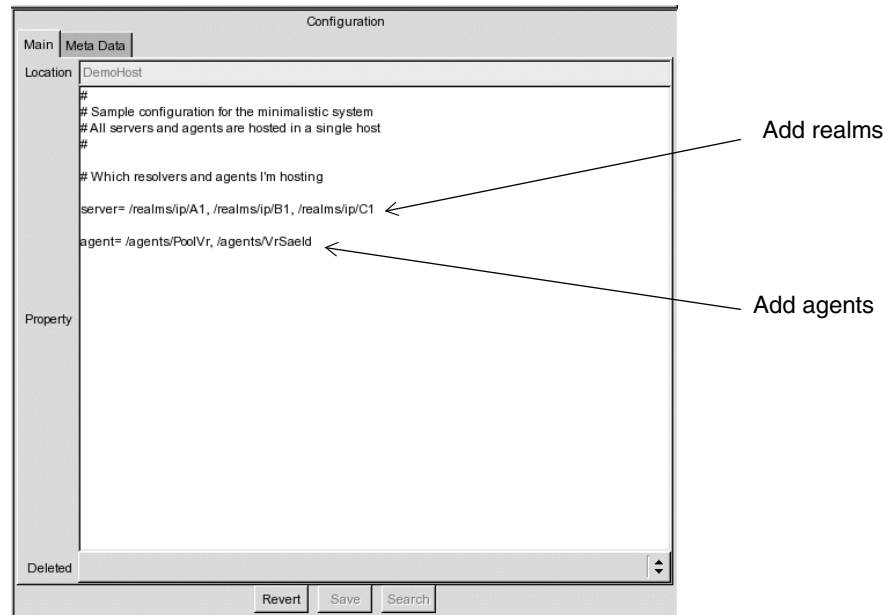
To use SDX Admin to customize a NIC configuration:

1. In the navigation pane, select a NIC configuration scenario under *l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc* to supply the basic type of resolution.
2. Expand the configuration scenario; then expand the agents item and the realms item for the configuration scenario, and review the configuration for each agent and each realm.
3. In the navigation pane, select a NIC configuration scenario under *l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc* that provides other types of resolutions that you want to add to the first configuration scenario.
4. Expand the configuration scenario; then expand the agents item and the realms item for the configuration scenario, and review the configuration for each agent and each realm.

5. Make a copy of the base resolution scenario:
 - a. Right-click the configuration scenario under *l = NIC*, *ou = staticConfiguration*, *ou = Configuration*, *o = Management*, *o = umc*, and select **Copy Tree**.
 - b. Right-click the *l = NIC* item, and select **Paste** (Configuration).
 - c. In the Paste Configuration dialog box, enter the name of the custom scenario in the Location box, and click **OK**.
 - d. Expand the configuration for the custom scenario and for the agents, hosts, and realms under that scenario.
6. Copy agents to be added to this configuration scenario from another configuration scenario. For each agent to be added:
 - a. Right-click the agent in the configuration scenario that is to be copied, and select **Copy Tree**.
 - b. Right-click the agent item under the configuration scenario, and select **Paste** (Configuration).
 - c. If the name of the agent does not appear in the list of agents, in the Paste Configuration dialog box keep the name of the Location the same, and click **OK**.

 If the name of the agent already appears in the list of agents and the configuration for this agent is different, in the Paste Configuration dialog box, enter a new name for the agent in the Location field, and click **OK**.
7. Copy each resolution to be added to the configuration scenario:
 - a. Right-click the resolution under *l = realms* in configuration scenario that is to be copied, and select **Copy Tree**.
 - b. Right-click *l = realms* under the new configuration scenario, and select **Paste** (Configuration).
 - c. In the Paste Configuration dialog box, keep the name of the location the same, and click **OK**.
8. If a non-Java application is to use the configuration scenario, copy the NIC locator configuration for each resolution:
 - a. Right-click the NIC locator under *l = nicLocators* in the configuration scenario that is to be copied, and select **Copy Tree**.
 - b. Right-click *l = nicLocators* under the new configuration scenario, and select **Paste** (Configuration).
 - c. In the Paste Configuration dialog box, keep the name of the location the same, and click **OK**.

9. Expand **hosts**, and select **DemoHost**. In the content pane, add entries for the agents added and the realms added.



10. Click **Save**.
11. If you make changes to a directory entry that result in the entry being removed from its search filter, stop and then restart the NIC host.

See *Managing Directory Changes for the Directory Agent* on page 295.

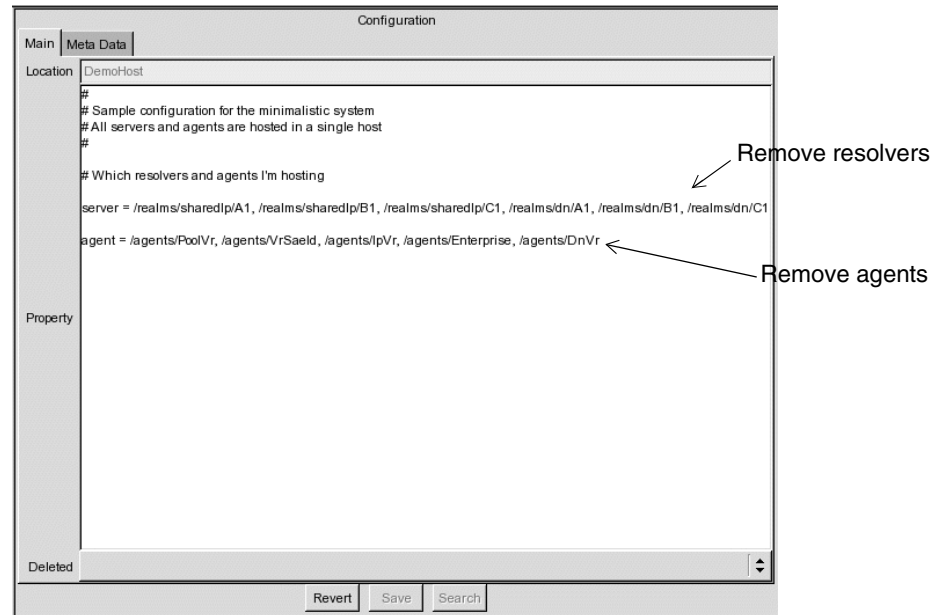
Creating a Custom NIC Configuration by Removing Components in an Existing Scenario

You can create a custom NIC configuration scenario in SDX Admin by copying an existing configuration and removing components from it. For example, you can remove IP resolution from the OnePopDnSharedIP scenario to create a CustomOnePopDn scenario.

To use SDX Admin to customize a NIC configuration by removing components:

1. In the navigation pane, select the NIC configuration scenario under *l = NIC, ou = staticConfiguration, ou = Configuration, o = Management, o = umc*.
2. Expand the configuration scenario; then expand the agents item and the realms item for the configuration scenario, and review the configuration for each agent and each realm.
3. To delete agents and resolvers that you want to remove, right-click the item and select **Delete**.

4. Expand **hosts**, and select **DemoHost**. In the content pane, remove entries for the agents and the resolvers that were deleted.



5. Click **Save**.
6. If you make changes to a directory entry that result in the entry being removed from its search filter, stop and then restart the NIC host.

See *Managing Directory Changes for the Directory Agent* on page 295.

Qualifying NIC Data Types

If a NIC configuration scenario uses the same data type in more than one place in a resolution, you can add an identifier to the data type to clarify the source of the data type.

You can qualify a NIC data type from:

- SDX Admin
- SDX Configuration Editor

Qualifying a NIC Data Type from SDX Admin

To use SDX Admin to qualify a data type:

1. In the navigation pane, select a resolution for a realm for a NIC configuration under *l = NIC, ou = staticConfiguration, o = Management, o = umc*.

For example; *l = ip, l = realms, l = OnePop, l = NIC, ou = staticConfiguration, o = Management, o = umc*.

2. Directly after a data type, add an identifier for the value by adding the value after the data type in parentheses.

In the following example, subscriber qualifies the IP data type on the transition.1 line, and cmts qualifies the IpPool data type on the transition.2 line.



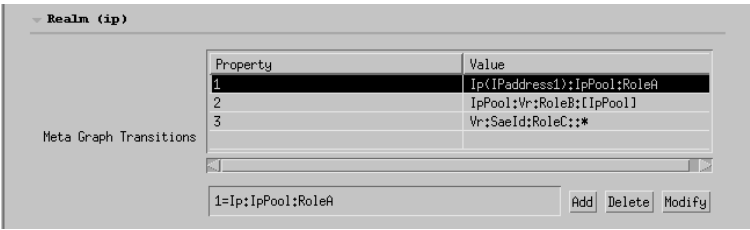
3. Click **Save**.

Qualifying a NIC Data Type from SDX Configuration Editor

To use SDX Configuration Editor to qualify a data type:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Realms** tab, and expand the **Realm < name >** section.
3. Select a row that lists a property and the associated value.

The property and value appear in the edit box below the list.

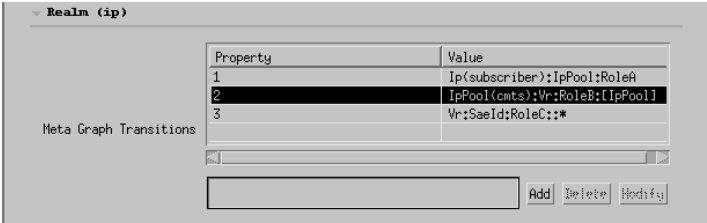


4. Add an identifier for the value by adding the value after the data type in parentheses, and click **Modify**.

For example, you would add each of the following lines separately:

```
IP(subscriber):IpPool:RoleA
IpPool(cmts):Vr:RoleB[IpPool]
```

After you edit an entry and then click **Modify** for each entry, the changes appear in the list.



5. Click **File > Save**.
6. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Managing Directory Changes for the Directory Agent

The NIC directory agent does not support dynamic changes to a directory entry that result in the entry's being removed from its search filter.

To have a NIC directory agent recognize a change in which a directory entry is removed from its search filter:

- Restart the NIC host that contains the agent.

For example, consider the MultiPop scenario provided as part of the NIC sample data. If you remove the POP-Ottawa scope from the directory entry with the following DN:

```
virtualRouterName=default, orderedCimKeys=Ottawa_ERX_Node, o=Network,  
o=umc
```

then the OttawaPoolVr and OttawaVrSaeId agents will not dynamically detect the change. You must restart OttawaHost for the changes to take effect.

Chapter 20

Reviewing the NIC Configuration

In most cases, you use the network information collector (NIC) configuration scenarios supplied in the sample data and make minor changes to the configuration to enable the configuration scenario to work in your environment.

For information about modifying NIC configuration scenarios for your use, see *Chapter 9, Locating Subscriber Information with the NIC*.

This chapter describes the NIC configuration as it appears in SDX Configuration Editor. Topics include:

- Reviewing the Configuration for NIC Realms on page 298
- Consolidator Agents on page 300
- Directory Agents on page 303
- SAE Plug-In Agents on page 307
- Properties Agents on page 313
- XML Agents on page 316
- Reviewing and Changing the Configuration for a NIC Host Instance on page 321
- Configuring Logging for NIC on page 322
- Reviewing the Configuration for NIC Locators on page 323
- Configuring Logging for NIC on page 322
- Reviewing the Configuration for NIC Locators on page 323
- Managing NIC Resolvers on page 324

Reviewing the Configuration for NIC Realms

NIC realms organize resolvers that perform a series of resolution tasks. Typically, you do not need to change the configuration for NIC realms.

To use SDX Configuration Editor to review the configuration for NIC realms:

1. In the navigation pane, select a configuration file for NIC.
2. Select the **Realms** tab, and expand the **Realm** section.

The screenshot shows the 'Realms' configuration interface. At the top, there's a 'Realms' header and a 'Realm' checkbox. Below this, there are two buttons: 'Create a New Instance of' with a dropdown menu currently showing 'Realm', and 'Delete an Instance'. A section titled 'Realm (test-realm)' is expanded, showing a table for 'Meta Graph Transitions'. The table has two columns: 'Property' and 'Value'. Below the table, there are three buttons: 'Add', 'Delete', and 'Modify'. At the bottom of the interface, there is a 'Resolvers' section with similar 'Create a New Instance of' and 'Delete an Instance' buttons.

3. Review the transitions (sequential resolutions) in the Meta Graph Transitions section.

The transition has the format property = value.

If you want to qualify a property or value in a transition:

- a. Select a transition.
- b. Edit the entry.
- c. Click **Modify**.

See *NIC Resolution Transition Fields* on page 299.

4. Expand the **Resolvers** section.

5. Review the configuration for resolvers to the realm.

See *NIC Resolvers Fields* on page 300.

NIC Resolution Transition Fields

In SDX Configuration Editor, you can view the transitions (sequential resolutions) that occur in the resolution process in the Realms section of the Realms pane in a NIC configuration file. The following fields define the transitions.

Property

- Sequential number of the transition
- Value—Integer
- Example—2
- Property name— < number >

Value

- Transitions that define the resolution process.
- Value—List of transitions, each of which has the format < key > ':' < value > ':' < role > ':' ('[' < 'constraint' > ']')* ('[' < constraint > ']')* ':' '*'
 - '—Literal value, for example ':' means a colon
 - < key > —Data type that is the key for the resolution process
 - < value > —Data type that is the value for the resolution process
 - < role > —Name of the resolver that handles this resolution process
 - ()—Optional elements

- **< constraint >**—Mandatory or optional condition that must or may be satisfied before the next stage of the resolution process can proceed. For dynamic constraints, **< constraint >** is the data type for the constraint (for example: IpPool). For static constraints, **< constraint >** has the format **< type > = < value > .**
 - ❑ **type**—Data type for the constraint.
 - ❑ **value**—Value to check against. Must be in the proper format expected by the data type (for example: Domain = virneo.com).
 - ❑ Append a '?' to the constraint if the constraint is optional.
 - ❑ *****—Operator that specifies that the resolver should accept the first response if it obtains information from multiple resolvers
- **Example**—2 = IpPool:Vr:L3:[IpPool][Domain?]

NIC Resolvers Fields

In SDX Configuration Editor, you can modify the following fields in the Resolvers section of the Realms pane in a NIC configuration file.

Resolver Role

- Role that this resolver executes.
- **Value**—Name of the role defined in the Meta Graph Transitions table at the top of the Realm section.
- **Example**—RoleA

Resolvers List

- Names of NIC resolvers to which this resolver forwards events.
- **Value**—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas

Roles List

- Names of NIC roles to which this resolver forwards events.
- **Value**—List of roles separated by commas
- **Example**—RoleA

Consolidator Agents

Consolidator agents are active NIC agents that publish data for passive agents, agents that provide information only on request. A consolidator agent comprises:

- A processor that takes a data mapping from a passive agent and returns either a network data object or a data mapping object that the consolidator agent then makes available.
- A component that tracks the number of times the processor adds or deletes an object. When the processor adds or deletes an object, this component publishes data for its associated resolvers.

Reviewing the Configuration of Consolidator Agents

To use SDX Configuration Editor to review the configuration for NIC consolidator agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **Consolidator Agent** section.

3. Review the entries in the fields.

See *Consolidator Agent Fields* on page 301.

Consolidator Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Consolidator Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format `<realmName> : <roleName> .`
 - `<realmName>` —Name of realm
 - `<roleName>` —Name of role

- Default—No value
- Property name—pushToRole

Source Agent

- Path to the agent for which this consolidator agent publishes data.
- Value—Text string
- Default—No value
- Example—/agents/InterfaceIdInterface
- Property name—sourceAgent

Agent Processor

- Name of the Java class that the NIC agent uses to generate the data value object.
- Value—Path to Java class
- Default—net.juniper.smgmt.gateway.nic.agent.dir consolidator.RouterEventProcessor
- Property name—processor.classname

Network Data Types

- Data types that the agent publishes; for names of data types, see *Chapter 18, NIC Resolution Process*.
- Value— <key> or <key> , <value>
 - <key> —Name of data key
 - <value> —Name of data value
- Default—No value
- Example—Ippool, InterfaceId
- Property name—NetworkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
 - 0—Data does not expire
 - Other values—Actual life expectancy of data

- Default—0
- Property name—eventLifeExpectancy

Directory Agents

Directory agents obtain information from the directory and are active NIC agents; they provide information by:

- Starting at a specified DN, reading directory entries that match the configured filter.
- Making available the directory entries that they read.
- Monitoring and publishing changes in the directory.

Reviewing the Configuration of Directory Agents

To use SDX Configuration Editor to review the configuration for NIC directory agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **Directory Agent** section.

The following sample Directory Agent section, shows a subset of the fields available for configuration.



Directory Agent (dir-agent-1)	
Resolvers List	<input type="text"/> <input type="button" value="Disable"/>
Roles List	<input type="text"/> <input type="button" value="Disable"/>
Search Base	<input type="text"/>
Search Filter	<input type="text"/> <input type="button" value="Disable"/>
Search Scope	<input type="text" value="SubTree"/> <input checked="" type="checkbox"/> <input type="button" value="Disable"/>
Server URL	<input type="text"/>

3. Review the entries in the fields.

See *Directory Agent Fields* on page 304.

Directory Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Directory Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—Comma-separated list of paths to NIC resolvers relative to the Static Configuration object.
- Default—No value
- Example—/realms/ip/B1
- Property name—pushtoServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format
 < realmName > : < roleName >
 - < realmName > —Name of realm
 - < roleName > —Name of role
- Default—No value
- Property name—pushToRole

Search Base

- DN of the location in the directory from which the agent should read information.
- Value— < DN > , < base >
- Default—No value
- Default—*o = Network, < base >*
- Property name—baseDN

Search Filter (optional)

- Directory search filter that the agent should use.
- Value—LDAP search filter
- Default—No value
- Example—(objectclass = umcVirtualRouter)
- Property name—searchFilter

Search Scope (optional)

- Location in the directory relative to the base DN from which the NIC agent can retrieve information.
- Value—One of the following options:
 - Object—Entry specified in the Search Base field only
 - Level—Entry specified in the Search Base field and objects that are subordinate by one level
 - Subtree—Subtree of entry specified in the Search Base field
- Default—Subtree
- Property name—searchScope

Server URL

- Location of the directory in URL string format.
- Value—Location of the directory that stores configuration information in URL string format `ldap:// <host> : <portNumber>`
 - <host> —IP address or name of directory host
 - <portNumber> —Number of TCP/IP port
- Default—No value
- Example—`ldap://127.0.0.1:389/`
- Property name—`java.naming.provider.url`

Backup Servers URL

- List of redundant directories.
- Value—List of URLs separated by semicolons
- Default—No value
- Example—`ldap://127.0.0.1:389/`
- Property name—`net.juniper.smgmt.des.backup_provider_urls`

Authentication DN

- DN that contains the username that the directory server uses to authenticate the NIC agent.
- Value—`<DN> , <base>`
- Default—No value
- Example—`cn = nic, ou = Components, o = Operators, <base>`
- Property name—`java.naming.security.principal`

Password

- Password that the directory server uses to authenticate the NIC agent.
- Value—`<password>`
- Guidelines—The password can be encoded in base64 and not visible in plain text. To use an encoded value, use the format `{BASE64} <encoded-value>`.

- Default—No value
- Example—nic
- Property name—java.naming.security.credentials

Key Attribute Name(s)

- Name of the directory attribute that the NIC agent uses for the network data object called *key*.
- Value—Name of an attribute in the directory
- Default—No value
- Example—virtualRouterName
- Property name—key.attrNames

Key Attribute Processor

- Java class that the NIC agent uses to generate the data key object.
- Value—Path to Java class
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.dir.DnAttributeProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- Directory attribute that the NIC agent uses for the network data object called *value*.
- Value—Name of an attribute in the directory
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Example—Saeld
- Property name—value.attrNames

Value Attribute Processor

- Name of the Java class that the NIC agent uses to generate the data value object.
- Value—Path to Java class
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Property name—net.juniper.smgt.gateway.nic.agent.dir.vr.VrAttributeProcessor

Network Data Types

- Names of the data types that this NIC agent publishes. For names of data types, see *Chapter 18, NIC Resolution Process*.
- Default—No value

- Example—IpPool,Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Example—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
 - 0—Data does not expire
 - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Directory Eventing Fields

For information about the directory eventing fields, see *SRC-PE Getting Started Guide, Chapter 32, Distributing Directory Changes to SRC Components on a Solaris Platform*.

SAE Plug-In Agents

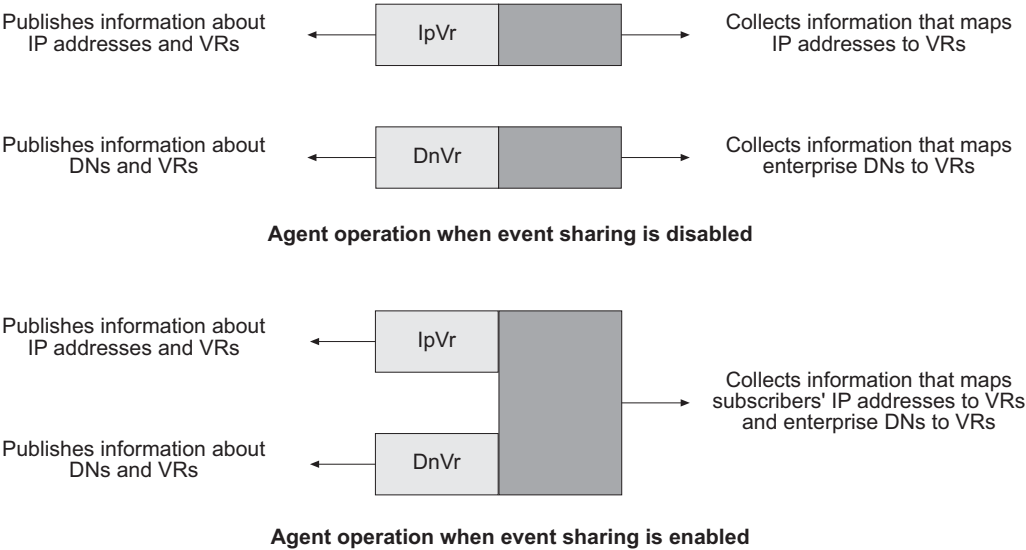
SAE plug-in agents collect information about the subscribers and interfaces managed by an SAE. For example, an SAE plug-in agent can collect and make available mappings of subscribers' IP addresses to the VRs from which those subscribers connect to the network. An SAE plug-in agent can also obtain information about the managed interfaces on a router.

We recommend that you have state synchronization enabled for SAE plug-in agents and that you use NIC replication to maintain high availability for the NIC. SAE plug-in agents always have information that is current unless the agent is overloaded and has not processed some of the SAE events; then the unprocessed events are placed in a queue.

You do not need to configure redundancy for the SAE plug-in agent as in past releases. If you have an SAE plug-in agent that uses agent redundancy, we recommend that you enable state synchronization for the agent and use NIC replication.

Each SAE plug-in agent is composed of two parts: one part that publishes NIC events and another part that collects SAE events. You can use multiple SAE plug-in agents supported on a single host to share the part that collects events. In this case, the SAE regards the agents as one plug-in, although the NIC still regards the agents as separate entities. Figure 12 illustrates this concept.

Figure 12: Event Sharing for SAE Plug-In Agents



9014661

For more information about SAE plug-ins, see *SRC-PE Subscribers and Subscriptions Guide, Chapter 8, Overview of Plug-Ins Included with the SAE*.

Reviewing the Configuration of SAE Plug-In Agents

To use SDX Configuration Editor to review the configuration for NIC SAE plug-in agents:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Agents** tab, and expand the **SAE Plug-In Agent** section.

Resolvers List	/realms/assignedIp/C1	Disable
Plug-in Event Type	Interface	Disable
Key Attribute Name(s)	PA_IF_INDEX, PA_ROUTER_NAME	
Key Attribute Processor	net.juniper.smgmt.gateway.nic.agent.saeplugin.Interface	Disable
Value Attribute Name(s)	PA_INTERFACE_NAME, PA_ROUTER_NAME	
Value Attribute Processor	net.juniper.smgmt.gateway.nic.agent.saeplugin.Interface	Disable
Naming Context	nicssae	
Event Filter		
Share the Event System	Yes	Disable
Enable State Synchronization	Yes	Disable
Number of Events Sent in a Synchronization Call	50	Disable
Event Database Filename	var/evdb	
Network Data Types	InterfaceId,Interface	
Event Life Expectancy	172800	Disable

3. Review the entries in the fields.

See *SAE Plug-In Agent Fields* on page 309.

SAE Plug-In Agent Fields

In SDX Configuration Editor, you can modify the following fields in the SAE Plug-In Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—Comma-separated list of paths to NIC resolvers relative to the Static Configuration object.
- Default—No value
- Example—/realms/dn/B1
- Property name—pushToServer

Plug-in Event Type

- Types of plug-in events that the agent supports.
- Value
 - User—Agent supports user-tracking plug-in events.
 - Interface—Agent supports interface-tracking plug-in events.
- Default—User
- Property name—pluginEventType

Key Attribute Name(s)

- List of plug-in attributes that provide information for the data key.
- Value—List of comma-separated plug-in attributes
- Guidelines—The list can contain one or more plug-in attributes.
If the format of the single plug-in attribute is not a string or you specify multiple plug-in attributes, the agent passes the data to the key processor to construct the data value in string format. In this case, you must specify the processor in the Key Attribute Processor field.
- Default—No value
- Example—PA_USER_DN, PA_ROUTER_NAME
- Property name—key.pluginAttributeNames

Key Attribute Processor

- Name of the Java class that the agent uses to generate the data key object.
- Value—Path to Java class
- Guidelines—Configure a key attribute processor if the agent acquires for the key value either a single plug-in attribute that is not in string format or multiple plug-in attributes.
- Default—No value
- Example—net.juniper.smgt.gateway.nic.agent.saeplugin.InterfaceIdProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- List of plug-in attributes that provide information for the data value.
- Value—List of comma-separated plug-in attributes
- Guidelines—The list can contain one or more plug-in attributes.
If the format of the single plug-in attribute is not a string or you specify multiple plug-in attributes, the agent passes the data to the value processor to construct the data value in string format. In this case, you must specify the processor in the Value Attribute Processor field.

- Default—No value
- Example—PA_USER_DN, PA_ROUTER_NAME
- Property name—value.pluginAttributeNames

Value Attribute Processor

- Name of the Java class that the agent uses to generate the data value object.
- Value—Path to Java class
- Guidelines—Configure a value attribute processor if the agent acquires for the data value either a single plug-in attribute that is not in string format or multiple plug-in attributes.
- Default—No value
- Example—net.juniper.smgmt.gateway.nic.agent.saeplugin.InterfaceProcessor
- Property name—value.processor.classname

Naming Context

- CORBA naming context in which the agent publishes references.
- Value—String that must match the context name in the objectref property for this SAE plug-in
- See *Chapter 18, NIC Resolution Process*.
- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Example—nicsaetestDNOttawa

This example matches the context name of the following objectref property:
corbaname::10.10.10.10:900/NameService#nicsaetestDNOttawa/saePort

- 10.10.10.10—Address of the machine running the CORBA naming server
- 900—TCP/IP port
- saePort—Name of plug-in (in this case, the agent eventing system)
- Property name—pluginNamingCtx

Event Filter

- LDAP filter that restricts the events that the agent collects.
- Value— < pluginAttribute > = < attributeValue >
 - < pluginAttribute > —Plug-in attribute name
 - < attributeValue > —Value of filter
- Default—No value
- Example—PA_USER_TYPE = INTF
- Property name—eventFilter

Share the Event System

- Specifies whether or not the agent shares the event system with other agents in the same host.
- Value
 - Yes—Agent shares the event system.
 - No—Agent does not share the event system.
- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Property name—eventSharingEnabled

Enable State Synchronization

- Specifies whether or not the state of the agent can be synchronized from the SAE. With state synchronization enabled, the state of the agent can be synchronized at any time.
- Value—Yes or No
- Guidelines—With state synchronization enabled, the agent uses NIC host redundancy.
- Default—Yes
- Property name—stateSyncEnabled

Number of Events Sent in a Synchronization Call

- Number of events the SAE sends to the agent at one time during state synchronization.
- Value—Integer in the range 1—2147483647
- Guidelines—This value is used if Enable State Synchronization is set to Yes.
- Default—50
- Property name—stateSyncBulkSize

Event Database Filename

- File in which the agent stores event information.
- Value—Path, relative to the directory that contains the NIC software, to the file
- Guidelines—If you configure event sharing for multiple SAE plug-in agents, this setting must be identical for all those agents.
- Default—No value
- Example—var/evdbDNMontral
- Property name—eventDbFilename

Network Data Types

- Attribute names for data that the agent collects in the format `< key > , < value >`
- Value
 - `< key >` —Attribute name for the data key
 - `< value >` —Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
 - 0—Data does not expire
 - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Properties Agents

A properties agent retrieves information from one or more specified property files and makes event information based on the information in the file available to the NIC. The format of the property file must comply with the typical format used for Java properties.

By default, property names in the file are keys, and property values are key values. If a property name (key) appears more than once in a file, the NIC uses the last value for the key in the file.

Although a properties agent may be used by an SRC application, typically you do not need to configure it. If you want to use a properties agent with a configuration scenario, contact Juniper Networks Professional Services.

Configuring Properties Agents

To use SDX Configuration Editor to configure properties agents:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Agents** tab, and expand the **Properties Agent** section.

3. Fill in the entries in the fields.

See *Properties Agent Fields* on page 314.

4. Select **File > Save**.
5. Right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

Properties Agent Fields

In SDX Configuration Editor, you can modify the following fields in the Properties Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Example— /realms/staticRouteIp/C1
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format `< realmName > : < roleName > .`
 - `< realmName >` —Name of realm
 - `< roleName >` —Name of role
- Default—No value
- Property name—pushToRole

Data Source

- List of URIs of property files that provides information about NIC events to the NIC system.
- Value—URIs separated by commas
- Guidelines—You must provide at least one URI.
- Default—No value
- Property name—dataSource

Network Data Types

- Attribute names for data that the agent collects in the format `< key > , < value >`
- Value
 - `< key >` —Attribute name for the data key
 - `< value >` —Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
 - 0—Data does not expire
 - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Reverse Values

- Specifies whether a property name is made available as a NIC key or a NIC value
- Value
 - Yes—Property names made available as keys
 - No—Property names made available as values
- Default—No value
- Property name—reverseValues

XML Agents

An XML agent retrieves information from a specified XML document and makes information available to NIC based on specified tags in the file. An XML agent provides information about one type of data or mappings.

Although an XML agent may be used by an SRC application, typically you do not need to configure it. If you want to use an XML agent with a configuration scenario, contact Juniper Networks Professional Services.

Configuring XML Agents

To use SDX Configuration Editor to configure XML agents:

1. In the navigation pane, select a NIC configuration file.
2. Select the **Agents** tab, and expand the **XML Agent** section.

The screenshot shows the 'Agents' pane in the SDX Configuration Editor. The 'Agent' section is expanded, showing a list of agents. The first agent is 'XML Agent (XML Agent 1)'. Below the agent list, there are various configuration fields for the XML Agent, each with a 'Disable' button. The fields are:

- Resolvers List
- Roles List
- Data Source
- Search Base
- Search Filter
- Search Scope (Set to 'SubTree')
- Element Types
- Key Attribute Name(s)
- Key Attribute Processor
- Value Attribute Name(s)
- Value Attribute Processor
- Network Data Types
- Publishing Interval (Set to '60')
- Event Life Expectancy (Set to '0')
- Enable Eventing (Set to 'Yes')

3. Modify the entries in the fields.

See *XML Agent Fields* on page 317.

4. Select **File > Save**.
5. right-click the configuration file, and select **SDX System Configuration > Export to LDAP Directory**.

XML Agent Fields

In SDX Configuration Editor, you can modify the following fields in the XML Agent section of the Agents pane in a NIC configuration file.

Resolvers List

- Names of NIC resolvers to which this agent sends events.
- Value—List of paths to NIC resolvers; paths are relative to the Static Configuration object and are separated by commas
- Default—No value
- Example— /realms/staticRouteIp/C1
- Property name—pushToServer

Roles List

- Names of NIC roles.
- Value—List of roles, separated by commas, in the format
 - < realmName > : < roleName >
 - < realmName > —Name of realm
 - < roleName > —Name of role
- Default—No value
- Property name—pushToRole

Data Source

- URI of the XML document that provides information about NIC events to the NIC system.
- Value— < URI >
- Guidelines—You must provide a URI for the XML document.
- Default—No value
- Property name—dataSource

Search Base

- Root XML element in the specified XML document at which the agent starts to search the XML document.
- Value— < XML element >
- If you do not specify an element for the search base, the agent starts searching at the top of the file.
- Default—No value
- Property name—base

Search Filter

- Search filter the agent uses to parse an XML document.
- Value—Search filter syntax defined in RFC 2254—The String Representation of LDAP Search Filters (December 1997)
- Default—No value

Search Scope

- Level at which the agent searches the XML document.
- Value
 - Object—Searches the object defined by the search base entry.
 - One level—Specifies objects at the same level as the object defined by the search base entry.
 - Subtree—Searches objects subordinate to object defined by the search base entry.
- Default—No value

Element Types

- Types of XML elements that the agent will use.
- Value—List of XML elements, separated by commas.
- Guidelines—Elements in this list must contain the attributes that the agent makes available.
- Default—No value
- Property name—elementTypes

Key Attribute Name(s)

- Name of the directory attribute that the NIC agent uses for the network data object called *key*.
- Value—Name of an attribute in the directory
- Default—No value
- Example—virtualRouterName
- Property name—key.attrNames

Key Attribute Processor

- Java class that the NIC agent uses to generate the data key object.
- Value—Path to Java class
- Default—No value
- Example—net.juniper.smgmt.gateway.nic.agent.dir.DnAttributeProcessor
- Property name—key.processor.classname

Value Attribute Name(s)

- Directory attribute that the NIC agent uses for the network data object called *value*.
- Value—Name of an attribute in the directory
- Guidelines—Specify only if the agent publishes mappings.
- Default—No value
- Example—Saeld
- Property name—value.attrNames

Value Attribute Processor

- Name of the Java class that the agent uses to generate the data value object.
- Value—Path to Java class
- Default—No value
- Property name—value.processor.classname

Network Data Types

- Attribute names for data that the agent collects in the format < key > , < value >
- Value
 - < key > —Attribute name for the data key
 - < value > —Attribute name for the data value
- Default—No value
- Example—Dn, Vr
- Property name—networkDataTypes

Publishing Interval

- Interval at which the NIC agent sends updates to the NIC resolvers.
- Value—Number of seconds in the range 0–2147483647
- Default—60
- Property name—publishingInterval

Event Life Expectancy

- Length of time that data is valid after the NIC proxy receives data associated with events published by this agent.
- Value—Number of seconds in the range 0–4294967295
 - 0—Data does not expire
 - Other values—Actual life expectancy of data
- Default—0
- Property name—eventLifeExpectancy

Enable Eventing

- Specifies whether or not the agent monitors changes to the XML document and sends events when changes occur.
- Value
 - Yes—Agent sends events.
 - No—Agent does not send events.
- Default—Yes
- Property name—enableEvents

Reviewing and Changing the Configuration for a NIC Host Instance

Typically, you use the DemoHost NIC host in the sample data. You review the configuration for the NIC host associated with a scenario on a system from SDX Configuration Editor.

To review the configuration for an instance of a NIC host:

1. Click the **Hosts** tab in a NIC configuration file.
2. Expand the Host entry.

The host details appear in the Hosts pane.

The screenshot shows a software interface titled "Hosts". It contains several expandable sections: "Logger", "Host", "Host (test_host)", and "Redundant Hosts". Each section has a "Create a New Instance of" button with a dropdown menu and a "Delete an Instance" button. The "Host" section's dropdown is set to "Host". The "Host (test_host)" section is expanded, showing two input fields: "Hosted Resolvers" and "Hosted Agents". The "Redundant Hosts" section also has "Create a New Instance of" and "Delete an Instance" buttons.

3. Review information about the NIC components that this host supports.

See *NIC Host Fields* on page 322.

NIC Host Fields

in SDX Configuration Editor, you can modify the following fields in the Host section of the Hosts pane in a NIC configuration file.

Hosted Resolvers

- Names of NIC resolvers that this host manages.
- Value—Comma-separated list of paths to NIC resolvers
 - Paths show the locations of the NIC resolvers relative to the Static Configuration object.
 - Subfolders in a path are separated by the forward slash (/).
- Example—/realms/sharedIp/A1, /realmssharedIp/B1, /realms/sharedIp/C1, /realms/dn/A1, /realms/dn/B1, /realms/dn/C1
- Property name—Server

Hosted Agents

- List of paths to NIC agents that this host supports.
- Value—Comma-separated list of paths to agents
 - Paths show the locations of the NIC agents relative to the Static Configuration object.
 - Subfolders in a path are separated by the forward slash (/).
- Example—/agents/VrSaeld, /agents/Router, /agents/PoolInterfaceId, /agents/InterfaceIdInterface
- Property name—Agent

Configuring Logging for NIC

You can configure one set of logging properties for all NIC hosts associated with a configuration. You can also configure a set of logging properties for each individual NIC host.

To use SDX Configuration Editor to configure logging for NIC:

1. In the navigation pane, select a NIC configuration file.
2. Click the **Hosts** tab.
3. To configure logging for individual NIC hosts, expand the **Logger** section subordinate to a host in the Hosts pane, and configure the fields.
4. To configure logging for all NIC hosts, expand **Logger** at the top of the Hosts pane, and configure the fields.

For information about logging properties and about managing log files, see:

- *SRC-PE Monitoring and Troubleshooting Guide, Chapter 3, Configuring Logging for SRC Components with the CLI*
- *SRC-PE Monitoring and Troubleshooting Guide, Chapter 4, Configuring Logging for SRC Components on a Solaris Platform*

Reviewing the Configuration for NIC Locators

Each NIC configuration scenario includes configuration for NIC locators for each type of resolution in the scenario. Non-Java applications communicate with NIC locators through the NIC access interface to perform data resolutions. Typically, you do not need to change the configuration for NIC locators. The configuration fields for NIC locators are the same as the configuration fields for NIC proxies.

To use SDX Configuration Editor to review the configuration for NIC locators:

1. Set the editing level for SDX Configuration Editor to Normal, Advanced, or Expert.
2. In the navigation pane, select a NIC configuration file.
3. Select the **NIC Locators** tab; expand the **NIC Locator Configuration** section, then the **NIC Locator** section, and then the **Resolution** section.

The screenshot shows the 'NIC Locator Configuration' dialog box. At the top, there are buttons for 'Create a New Instance of' and 'Delete an Instance'. Below these, the 'NIC Locator (ip)' section is expanded, showing the 'Resolution' sub-section. The 'Resolution' section contains the following fields:

- Resolver Name:** /realms/ip/R1
- Key Type:** Ip
- Value Type:** SAE server ID
- Expect Multiple Values:** No
- Constraints:** (empty field)
- Use Local NIC Host:** (checked)
- Cache Size:** 0

A 'Disable' button is located at the bottom right of the dialog.

4. Review the entries in the fields.

The configuration fields for a NIC locator are the same as a subset of the fields for a NIC proxy.

For information about NIC locator configuration fields, see the description of the NIC proxy fields in *Chapter 13, Configuring Applications to Communicate with an SAE*.

Managing NIC Resolvers

You can also view data mappings for NIC resolvers with a program called **nicDump**, which is installed in the folder `/opt/UMC/nic/bin` when you install a NIC host on a machine.

When you run **nicDump**, you can view information for all resolvers that the NIC host supports or information about a specific NIC resolver. The program writes the output to the file you specify or to the default file, `/opt/UMC/nic/var/dumpResult.txt`. The syntax for the command is:

```
nicDump [ [ -r <resolverName> ] [-f <fileName> ] ] --help ]
```

- `<resolverName>` —Name of the resolver for which you want to view information in the specified file; the value is case sensitive and must match the name in the NIC configuration.
- `<fileName>` —Name of the file in which you want to place state information for the specified resolver; the value is case sensitive.

To generate information about NIC resolvers with **nicDump**, run the program.

Example 1 `# /opt/UMC/nic/bin/nicDump -r /realms/ip/A1 -f /opt/UMC/nic/A1_data.txt`
`# more /opt/UMC/nic/A1_data.txt`

```
##### DUMP OUTPUT ##### Wed Sep 24 11:01:00 EDT 2003
Resolver = /realms/ip/A1
[lpPool:([10.227.25.208 10.227.25.255]), lpPool:([10.227.1.1
10.227.255.255]), lpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.
255])]
```

Example 2 `# /opt/UMC/nic/bin/nicDump`
`# more /opt/UMC/nic/var/dumpResult.txt`

```
##### DUMP OUTPUT ##### Tue Sep 23 13:22:32 EDT 2003
[lpPool:([10.227.25.208 10.227.25.255]), lpPool:([10.227.1.1
10.227.255.255]), lpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255])]
[pair<lpPool:([10.227.25.208 10.227.25.255]), Vr:default@phoenix>,
pair<lpPool:([10.227.1.1 10.227.255.255]), Vr:default@bigfoot>,
pair<lpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255]), Vr:default@erx-node1>]
```

```
[pair<Vr:default@erx-node1, Saeld:IOR:12345>, pair<Vr:default@bigfoot,
Saeld:IOR:8888888888>]
```

```
##### DUMP OUTPUT ##### Tue Sep 23 13:58:48 EDT 2003
[lpPool:([10.227.25.208 10.227.25.255]), lpPool:([10.227.1.1
10.227.255.255]), lpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255])]
[pair<lpPool:([10.227.25.208 10.227.25.255]), Vr:default@phoenix>,
pair<lpPool:([10.227.1.1 10.227.255.255]), Vr:default@bigfoot>,
pair<lpPool:([10.10.1.0 10.10.255.255][10.20.0.0
10.20.255.255][10.227.1.100 10.227.1.255]), Vr:default@erx-node1>]
[pair<Vr:default@erx-node1, Saeld:IOR:12345>, pair<Vr:default@bigfoot,
Saeld:IOR:8888888888>]
```


Chapter 21

NIC Configuration Scenarios

This chapter provides detailed descriptions of the network information collector (NIC) configuration scenarios. Topics include:

- Overview of NIC Configuration Scenarios on page 328
- OnePop Scenario on page 328
- OnePopPcmm Scenario on page 332
- OnePopDynamicIp Scenario on page 334
- OnePopSharedIp Scenario on page 336
- OnePopStaticRouteIp on page 338
- OnePopAcctId Scenario on page 340
- OnePopLogin Scenario on page 342
- OnePopPrimaryUser on page 345
- OnePopDnSharedIp Scenario on page 347
- OnePopAllRealms Scenario on page 351
- MultiPop Scenario on page 355

Overview of NIC Configuration Scenarios

The NIC configuration scenarios in the sample data provide resolutions for a variety of network configurations.

Each NIC scenario includes two types of configuration:

- Centralized—A single host configuration for use with NIC replication. In a centralized configuration all agents and resolvers reside on one host. The name of this host is DemoHost.
- Distributed—A multiple host configuration in which agents and resolvers are distributed among more than one host. This type of configuration is designed for use with NIC host redundancy. In most cases, the hosts are named OnePopH1 (a host in a pop) and OnePopBO (a host in a back office).

The best way to view the sample data is with the NIC Web Admin tool.

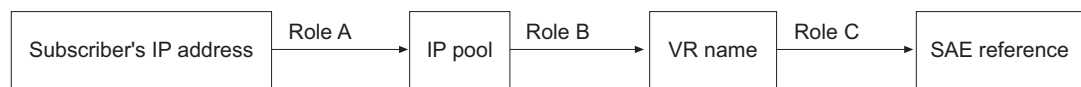
For a summary of the NIC configuration scenarios included in the sample data, see *Chapter 9, Locating Subscriber Information with the NIC*.

OnePop Scenario

The OnePop scenario illustrates a configuration that supports one POP. The realm for this configuration accommodates the situation in which IP address pools are configured locally on each VR. The resolution process takes a subscriber's IP address as the key and returns a reference to the SAE managing this subscriber as the value.

Figure 13 shows the resolution graph for this realm.

Figure 13: Resolution Process for ip Realm



g014923

The following agents collect information for resolvers in this realm:

- Directory agent PoolVr collects and publishes information about the mappings of IP address pools to VRs.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs.

The OnePop sample provides two host configurations: a centralized configuration and a distributed configuration. The OnePop Centralized configuration also provides an example of NIC host redundancy.

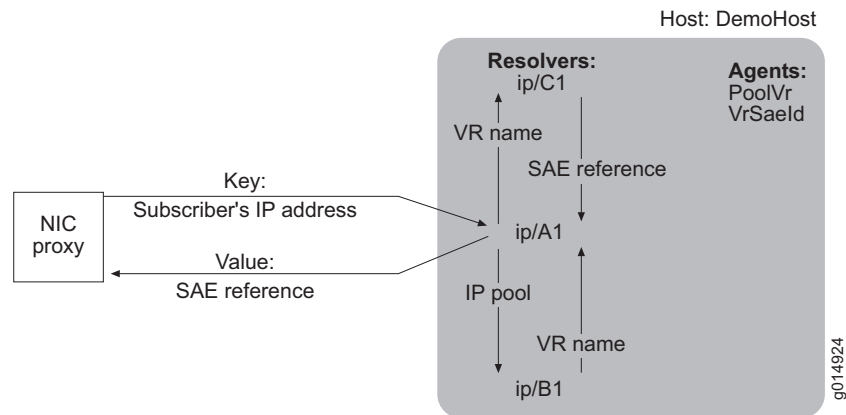
Centralized Configuration

In this configuration, single host DemoHost supports all agents and resolvers. When the NIC proxy sends a subscriber's IP address to host DemoHost, the following sequence of actions occurs:

1. The host passes the IP address to resolver A1.
2. Resolver A1 obtains an IP pool for the IP address and forwards the request to resolver B1.
3. Resolver B1 obtains a VR name for the IP pool and returns the VR name to resolver A1.
4. Resolver A1 forwards the VR name to resolver C1.
5. Resolver C1 obtains an SAE reference for the VR and returns the VR identity to resolver A1.
6. Resolver A1 passes the SAE reference to its host.
7. The host returns the SAE reference to the NIC proxy.

Figure 14 shows the interactions of the NIC components for this realm.

Figure 14: OnePop Centralized Configuration

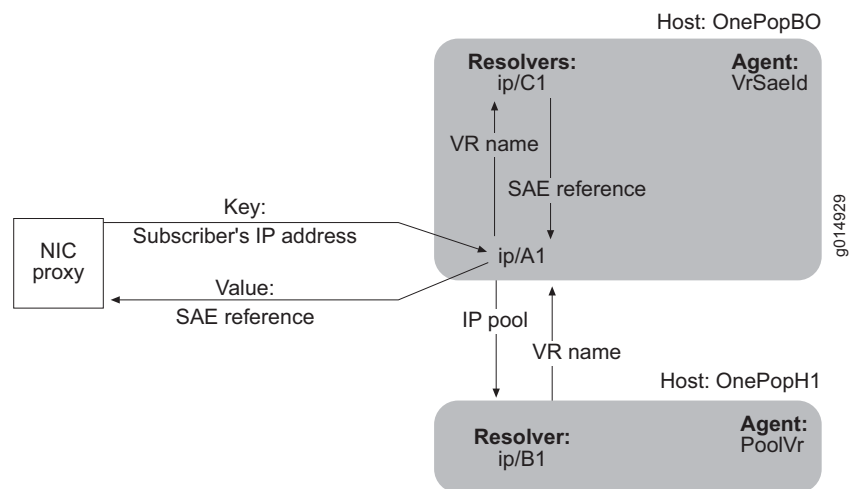


Distributed Configuration

In this configuration, the agents and resolvers are distributed among several hosts. When the NIC proxy sends a subscriber's IP address to host OnePopBO, the components execute the same actions as they do in the centralized configuration (see *Centralized Configuration* on page 329).

Figure 15 illustrates the interactions of the NIC components for this realm.

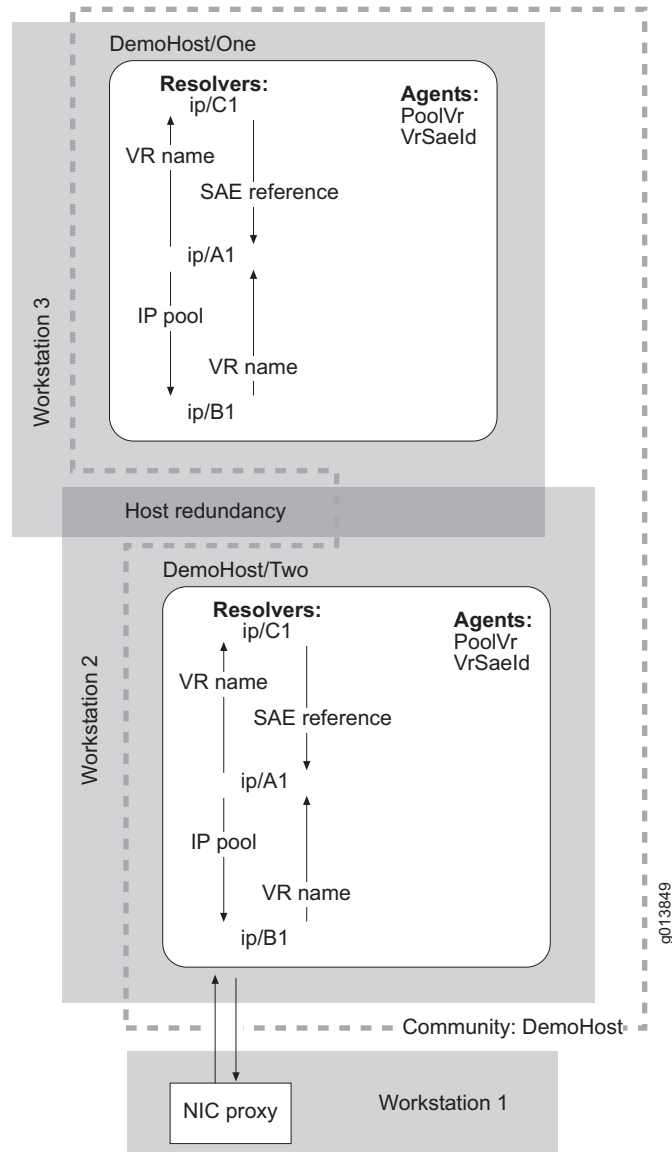
Figure 15: OnePop Distributed Configuration



Redundancy

This sample data includes host redundancy for the centralized configuration. The hosts DemoHost/One and DemoHost/Two, which are installed on different machines, provide host redundancy. These hosts form the community DemoHost, which does not include a monitor.

Figure 16: Redundancy for OnePop Centralized Configuration



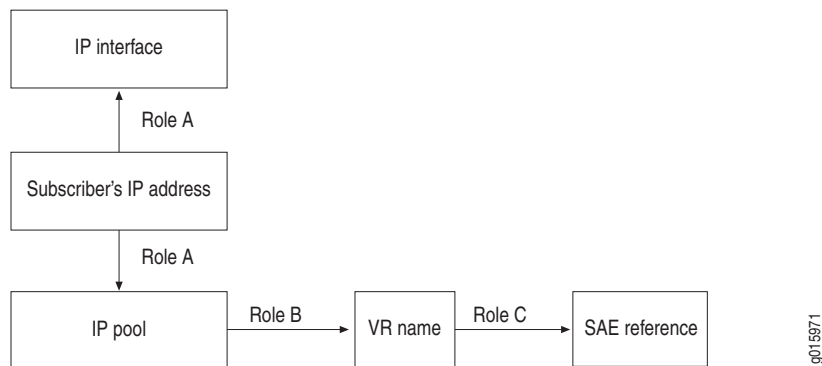
OnePopPcmm Scenario

This scenario is similar to the OnePop configuration scenario. It illustrates a configuration in which an assigned subscriber IP address managed by a network device such as a cable modem termination system (CMTS) device resolves to a reference to the SAE managing this subscriber. In this situation, the SAE acts as an application manager and interacts with the CMTS through a policy server.

The OnePopPcmm configuration scenario supports a PacketCable Multimedia Specification (PCMM) environment in which you use the assigned IP subscriber method to log in subscribers and in which you use the NIC to determine the subscriber's SAE. The realm for this configuration accommodates the situation in which IP pools are configured locally on each application manager group object. These IP pools represent an IP pools-managed policy decision point (PDP) group for one or more CMTS devices.

Figure 17 shows the resolution graph for this realm.

Figure 17: Resolution Process for Pcmm_am Realm



This scenario uses the same agents as the OnePop scenario. For the OnePopPcmm configuration scenario, the agent collects information from the application manager object instead of the virtual router entry. A virtual router name is generated in the format "default"@ < pdpGroup > .

The OnePopPcmm scenario provides two host configurations: a centralized configuration and a distributed configuration.

Centralized Configuration

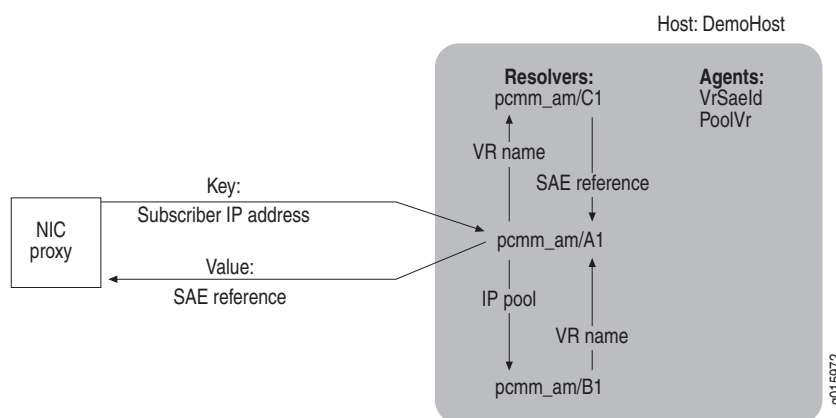
In this configuration, the single host DemoHost supports all agents and resolvers. When a NIC proxy sends a subscriber's IP address to host DemoHost, the following sequence of actions occurs:

1. The host passes an assigned subscriber IP address resolver A1.
2. Resolver A1 obtains the IP pool name and the interface name, and forwards the request to resolver B1.
3. Resolver B1 obtains the VR name for the IP pool name and interface name, and returns the VR name to resolver A1.

4. Resolver A1 forwards the VR name to resolver C1.
5. Resolver C1 obtains an SAE reference for the VR and returns it to resolver A1.
6. Resolver A1 passes the SAE reference to its host.
7. The host returns the SAE reference to the NIC proxy.

Figure 18 show the interactions of the NIC components for this realm.

Figure 18: OnePopPcmm Centralized Configuration

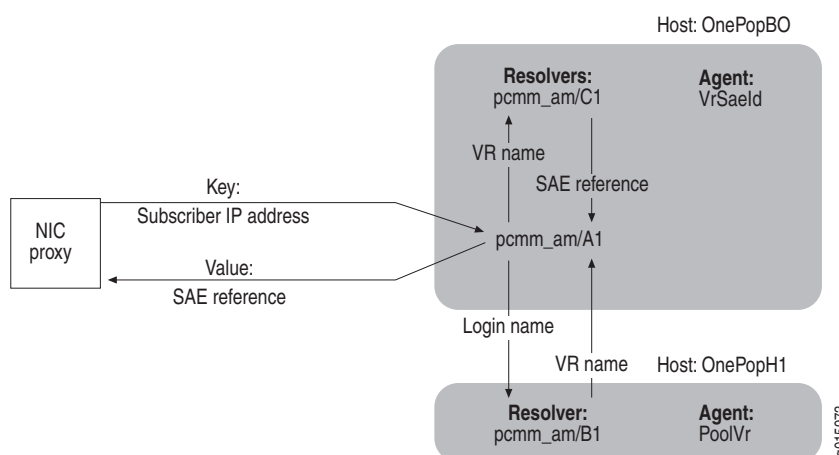


Distributed Configuration

In this configuration, the agents and resolvers are distributed among two hosts. When the NIC proxy sends a subscriber's IP address to host OnePopBO, the components execute the same actions as they do in the centralized configuration (see *Centralized Configuration* on page 332).

Figure 19 illustrates the interactions of the NIC components for this realm.

Figure 19: OnePopPcmm Distributed Configuration



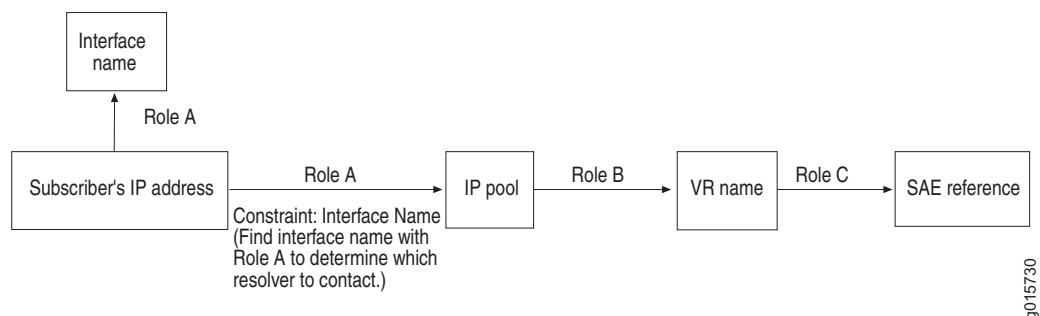
OnePopDynamicIp Scenario

This scenario illustrates a configuration that is very similar to the OnePop scenario. The realm for this configuration accommodates the situation in which IP address pools are configured locally on each virtual router object. The resolution process takes a subscriber's IP address as the key and returns a reference to the SAE managing this subscriber as the value.

The scenario supports a configuration scenario for a PacketCable Multimedia Specification (PCMM) environment in which you use the assigned IP subscriber method to log in subscribers, and use the NIC to determine the subscriber's SAE. In this scenario, the SAE acts as a combined application manager and policy server; it directly manages CMTS devices.

Figure 20 shows the resolution graph for this realm.

Figure 20: Resolution Process for dynamicIp Realm



The following agents collect information for resolvers in this realm:

- Directory agent PoolVr collects and publishes information about the mappings of IP address pools to VRs.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs.

The OnePopDynamicIp scenario provides two host configurations: a centralized configuration and a distributed configuration.

Centralized Configuration

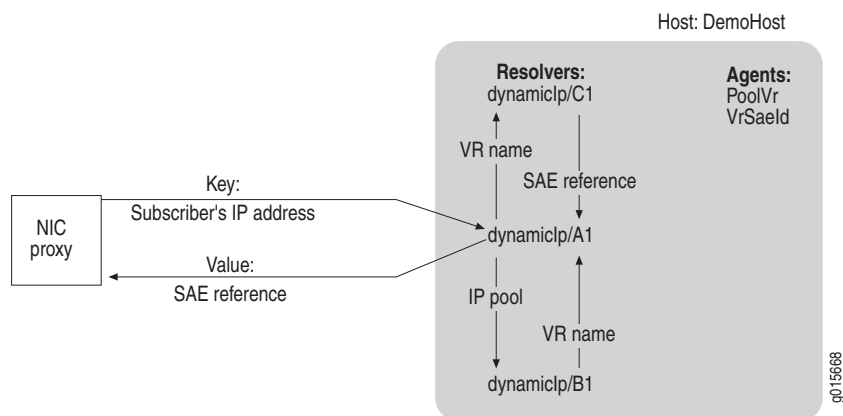
In this configuration, single host DemoHost supports all agents and resolvers. When the NIC proxy sends a subscriber's IP address to host DemoHost, the following sequence of actions occurs:

1. The host passes the IP address to resolver A1.
2. Resolver A1 obtains an IP pool name and interface name for the IP address, and forwards the request to resolver B1.
3. Resolver B1 obtains a VR name for the IP pool name and interface name, and returns the VR name to resolver A1.
4. Resolver A1 forwards the VR name to resolver C1.

5. Resolver C1 obtains an SAE reference for the VR and returns the VR identity to resolver A1.
6. Resolver A1 passes the SAE reference to its host.
7. The host returns the SAE reference to the NIC proxy.

Figure 21 illustrates the interactions of the NIC components for this realm.

Figure 21: OnePopDynamicIp Centralized Configuration

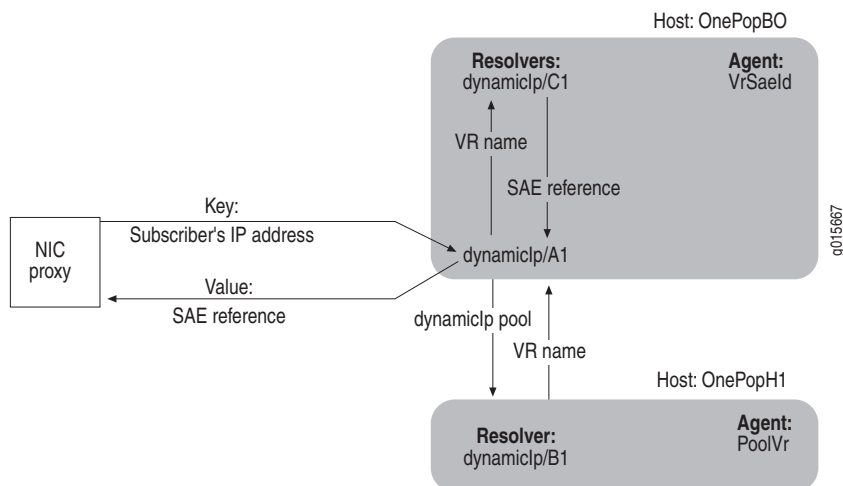


Distributed Configuration

In this configuration, the agents and resolvers are distributed among several hosts. When the NIC proxy sends a subscriber's IP address to host OnePopBO, the components execute the same actions as they do in the centralized configuration (see *Centralized Configuration* on page 334).

Figure 22 illustrates the interactions of the NIC components for this realm.

Figure 22: OnePopDynamicIp Distributed Configuration

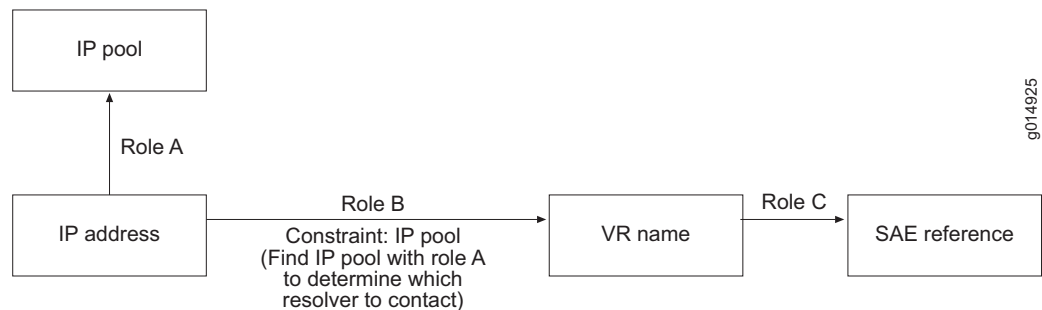


OnePopSharedIp Scenario

This scenario illustrates a configuration that is very similar to the OnePop scenario. However, the realm for this configuration accommodates the situation in which IP address pools are shared by VRs in the same POP. The resolution process takes a subscriber's IP address as the key and returns a reference to the SAE managing this subscriber as the value.

Figure 23 shows the resolution graph for this realm.

Figure 23: Resolution Process for sharedIp Realm



The following agents interact with resolvers in this realm:

- SAE plug-in agent IpVr collects and publishes information about the mappings of IP addresses to VRs.
- Directory agent PoolVr collects and publishes information about the IP address pools used by the VRs in a POP. Because the IP address pools are shared between VRs, this agent discards information about VRs.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs.

The OnePopSharedIP scenario provides two host configurations: a centralized configuration and a distributed configuration.

Centralized Configuration

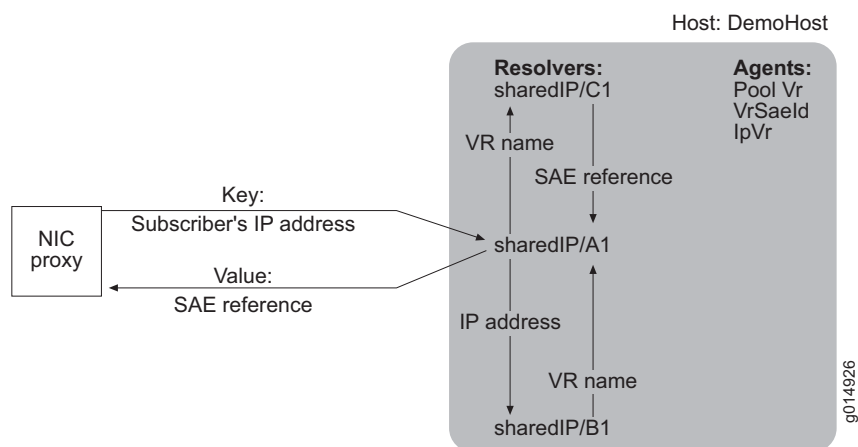
In this configuration, single host DemoHost supports all agents and resolvers. When the NIC proxy sends a subscriber's IP address to host DemoHost, the following sequence of events occurs:

1. The host passes the IP address to resolver A1.
2. Resolver A1 obtains an IP pool for the IP address.
3. Resolver A1 forwards the IP address and the IP pool to resolver B1.
4. Resolver B1 obtains a VR name for the IP address and returns the VR name to resolver A1.
5. Resolver A1 forwards the VR name to resolver C1.

6. Resolver C1 obtains an SAE reference for the VR and returns the SAE reference to resolver A1.
7. Resolver A1 passes the SAE reference to its host.
8. The host returns the SAE reference to the NIC proxy.

Figure 24 shows the interactions of the NIC components for this realm.

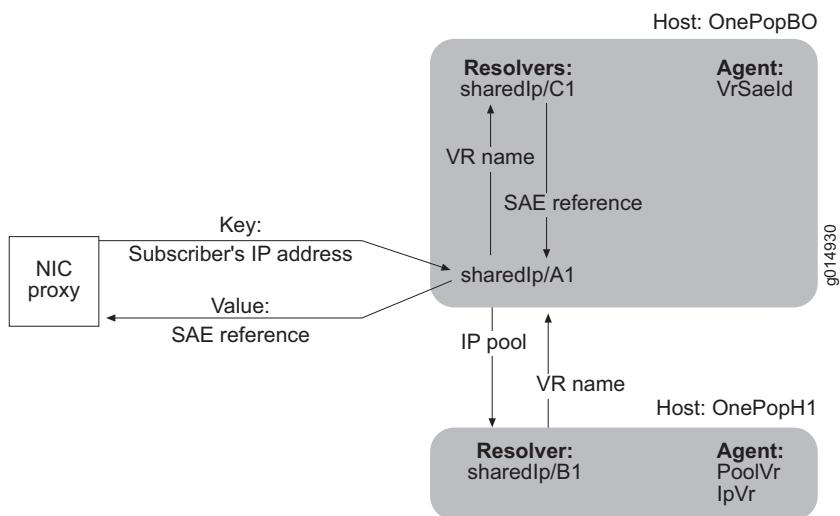
Figure 24: OnePopSharedIP Centralized Configuration



Distributed Configuration

In this configuration, the agents and resolvers are distributed among several hosts. When the NIC proxy sends a subscriber's IP address to the host OnePopBO, the resolvers execute the same actions as they do in the centralized configuration. Figure 25 illustrates the interactions of the NIC components for this realm.

Figure 25: OnePopSharedIP Distributed Configuration

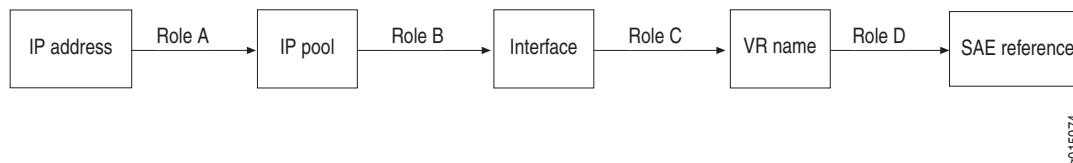


OnePopStaticRouteIp

The OnePopStaticRouteIp configuration scenario for NIC resolves an assigned IP address for a subscriber whose traffic enters the network through an interface on a JUNOS routing platform to a reference for the SAE that manages the interface. The realm for this configuration accommodates the situation in which the network publisher component gathers interface information for the JUNOS routing platforms. The resolution process takes a subscriber's IP address as a key and returns a reference to the SAE that manages the interface.

Figure 26 shows the resolution graph for this realm.

Figure 26: Resolution Process for the StaticRouteIp Realm



The following agents collect information for resolvers in this realm:

- Directory agent PoolInterface collects and publishes information about the mappings of IP address pools to interfaces.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs.

The agents obtain information from the interfaceConfiguration attribute of the EdgeRouter entry in the directory and read an XML document that conforms to the networkConfig.xsd schema. If this scenario is used with a different router type, you can edit the XML document.

For information about the XML document, see *Chapter 12, Obtaining Interface Configuration for OnePopStaticRouteIp*.

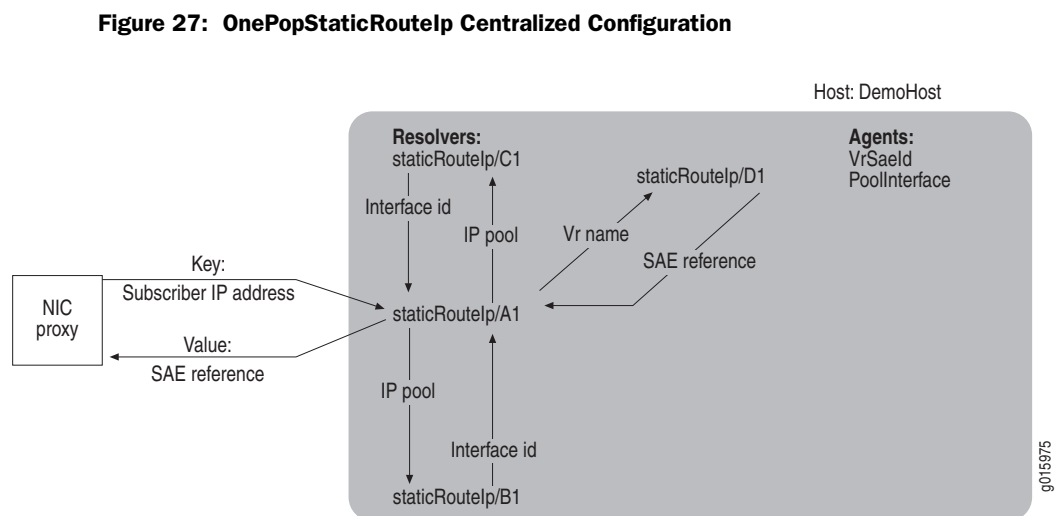
The OnePopStaticRouteIp scenario provides two host configurations: a centralized configuration and a distributed configuration.

Centralized Configuration

In this configuration, the single host DemoHost supports all agents and resolvers. When the NIC proxy sends a subscriber's IP address to host DemoHost, the following sequence of events occurs:

1. The host passes the subscriber's IP address to resolver A1.
2. Resolver A1 obtains an IP pool for the IP address.
3. Resolver A1 forwards the IP pool name to Resolver B1.
4. Resolver B1 obtains the interface ID for the IP pool and returns this value to resolver A1.
5. Resolver A1 forwards the interface ID to Resolver C1.
6. Resolver C1 resolves the interface ID to the VR name and returns the VR name to resolver A1.
7. Resolver A1 forwards the VR name to resolver D1.
8. Resolver D1 obtains a reference for the SAE managing the VR and returns the SAE reference to resolver A1.
9. Resolver A1 passes the SAE reference to its host.
10. The host returns the SAE reference to the NIC proxy.

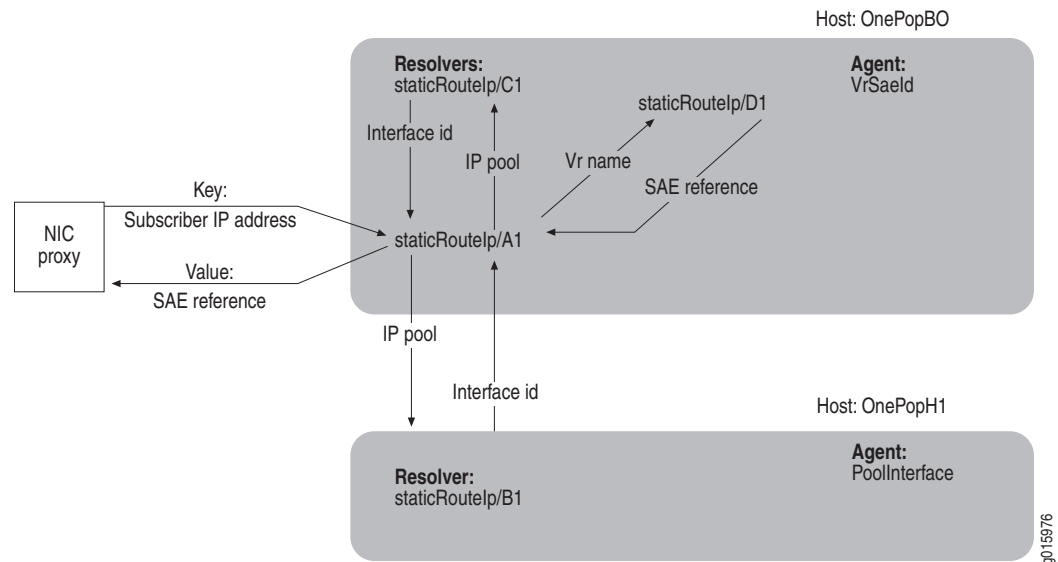
Figure 27 shows the interactions of the NIC components for this realm.



Distributed Configuration

In this configuration, the agents and resolvers are distributed among two hosts. When a NIC proxy sends a subscriber IP address to host OnePopBO, the resolvers execute the same actions as they do in the centralized configuration. Figure 28 illustrates the interactions of the NIC components for this realm.

Figure 28: OnePopStaticRouteIp Distributed Configuration

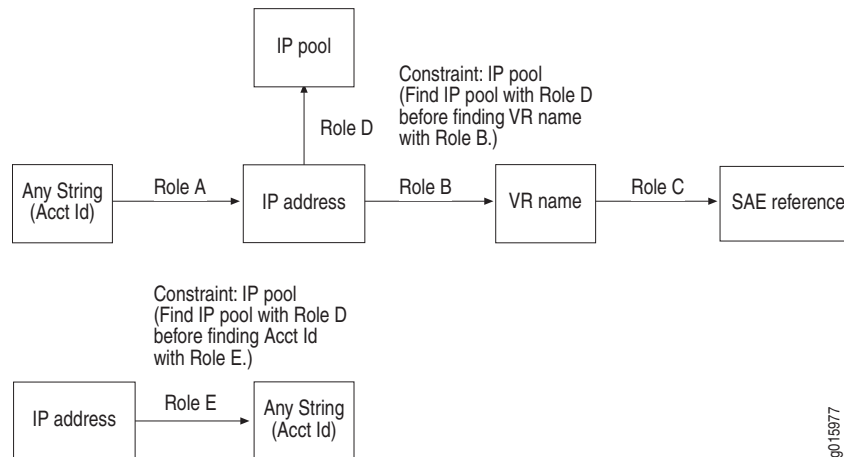


OnePopAcctId Scenario

This scenario illustrates a configuration in which subscribers have an accounting ID, as defined by the LDAP attribute `accountingUserId` or the plug-in attribute `PA_ACCOUNTING_ID`. The realms for this configuration accommodate two independent resolution processes, which can be used by the SRC Volume-Tracking Application (SRC-VTA).

Figure 29 shows the resolution graphs for this realm.

Figure 29: Resolution Process for acctId Realm



The following agents collect information for resolvers in this realm:

- Directory agent PoolVr collects and publishes information about the mappings of IP address pools to VRs.
- Directory agent VrSaeld collects and publishes information about the mappings of virtual routers and the mappings between virtual routers and SAEs.
- SAE plug-in agent AcctIdIp collects and publishes information about the mappings of accounting IDs of subscribers to subscriber IP addresses.
- SAE plug-in agent IpAcctId collects and publishes information about the mappings of subscriber IP addresses to accounting IDs.

The OnePopAcctId scenario provides one host for a centralized configuration. In this configuration the single host DemoHost supports all agents and resolvers. Two NIC proxies are associated with the configuration. One NIC proxy (called acct-sae in this description) submits accounting IDs, and another NIC proxy (called addr-acct in this description) submits subscribers' IP addresses.

When the NIC proxy sends an accounting ID to host DemoHost, the following sequence of events occurs:

1. The host passes the subscriber's accounting ID to resolver A1.
2. Resolver A1 obtains an IP address for the account ID.
3. Resolver A1 forwards the IP address to Resolver D1.
4. Resolver D1 obtains the IP pool for the IP address and returns it to Resolver A1.
5. Resolver A1 forwards the IP address and IP pool to Resolver B1.
6. Resolver B1 obtains the VR name and return it to resolve A1.

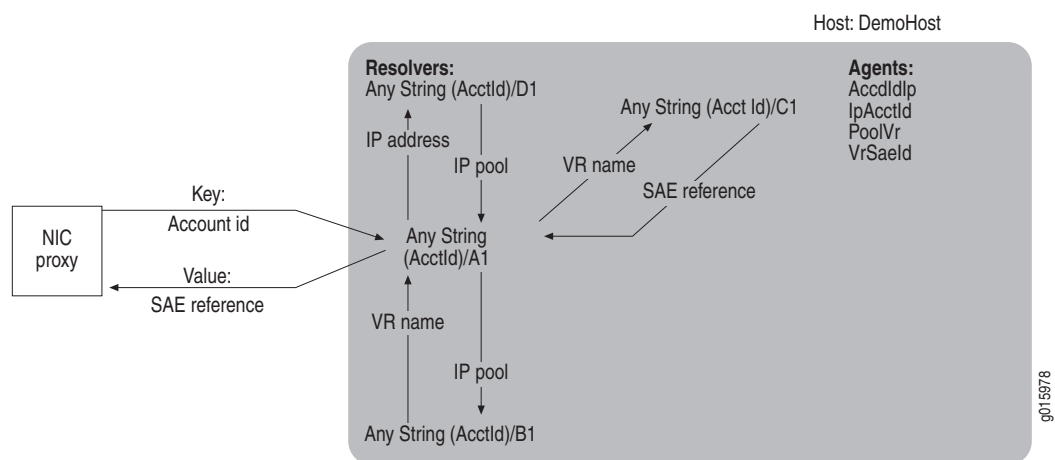
7. Resolver A1 forwards the VR name to resolver C1.
8. Resolver C1 obtains the SAE reference for the VR name and returns it to resolver A1.
9. Resolver A1 passes the SAE reference to its host.
10. The host returns the SAE reference to the NIC proxy acct-sae.

When the NIC proxy sends an IP address to host DemoHost, the following sequence of events occurs:

1. The host passes the subscriber's IP address to resolver A1.
2. Resolver A1 forwards the IP address to resolver D1.
3. Resolver D1 obtains the IP pool for the IP address and returns it to resolver A1.
4. Resolver A1 forwards the IP address and IP pool to resolver C1.
5. Resolver C1 obtains the accounting ID for the IP address and associated IP pool and returns the accounting ID to resolver A1.
6. Resolver A1 passes the accounting ID to its host.
7. The host returns the accounting ID to the NIC proxy addr-acct.

Figure 30 illustrates the interactions of the NIC components for this realm.

Figure 30: OnePopAcctId Centralized Configuration

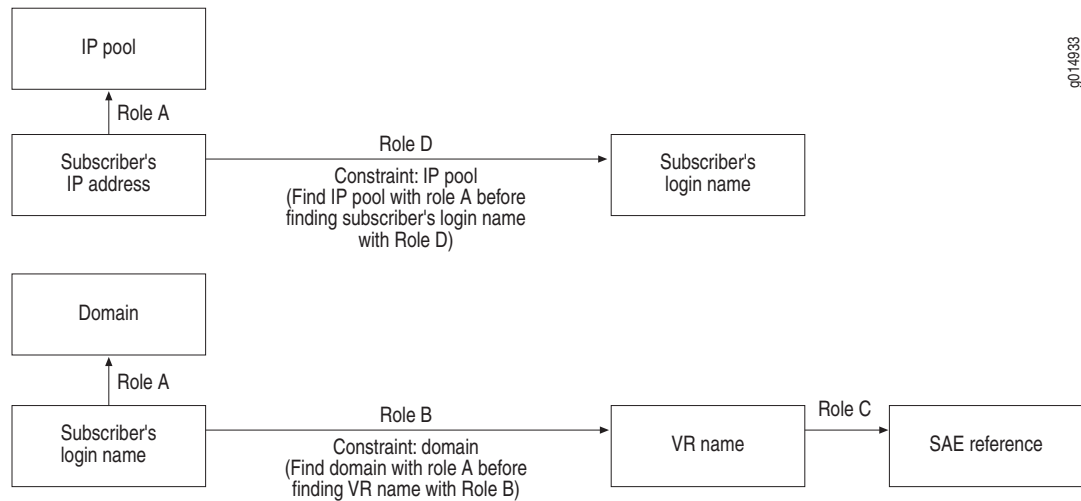


OnePopLogin Scenario

This scenario illustrates a configuration that is very similar to the OnePop scenario. The realm for this configuration accommodates two independent resolution processes, which are used by the SRC-Volume-Tracking Applications (SRC-VTAs) and may be used for other purposes.

Figure 31 shows the resolution graphs for this realm.

Figure 31: Resolution Processes login Realm



The following agents interact with resolvers in this realm:

- SAE plug-in agent IpLoginName collects and publishes information about the mappings of IP addresses to login names.
- SAE plug-in agent LoginNameVr collects and publishes information about the mappings of login names to VRs.
- Directory agent Pool collects and publishes information about the IP address pools used by the VRs in a POP. The agent uses the information about the IP address pools to determine which resolver to communicate with, rather than communicating with all resolvers that are running role D.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs.

The OnePopLogin scenario provides two host configurations: a centralized configuration and a distributed configuration.

Centralized Configuration

In this configuration, single host DemoHost supports all agents and resolvers. Two NIC proxies are associated with this NIC configuration; one NIC proxy (called NIC proxy 1 in this documentation) submits subscribers' login names, and the other (called NIC proxy 2 in this documentation) submits subscribers' IP addresses.

When NIC proxy 1 sends a login name to the host DemoHost, the following sequence of events occurs:

1. The host passes the login name to resolver A1.
2. Resolver A1 obtains a domain name for the login name.

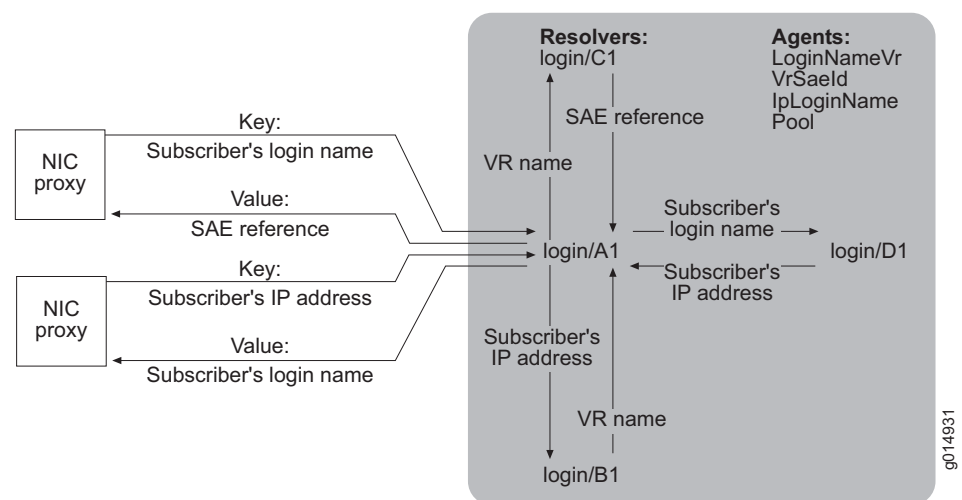
3. Resolver A1 forwards the login name and the domain to resolver B1.
4. Resolver B1 obtains a VR name for the login name and returns the VR name to resolver A1.
5. Resolver A1 forwards the VR name to resolver C1.
6. Resolver C1 obtains an SAE reference for the VR and returns the SAE reference to resolver A1.
7. Resolver A1 returns the SAE reference to its host.
8. The host returns the SAE reference to the NIC proxy.

When NIC proxy 2 sends a subscriber's IP address to host DemoHost, the following sequence of events occurs.

1. The host passes the IP address to resolver A1.
2. Resolver A1 obtains an IP pool for the IP address.
3. Resolver A1 forwards the IP address and the IP pool to resolver D1.
4. Resolver D1 obtains a login name for the IP address and returns the login name to resolver A1.
5. Resolver A1 passes the login name to its host.
6. The host returns the login name to the NIC proxy.

Figure 32 illustrates the interactions of the NIC components for this realm.

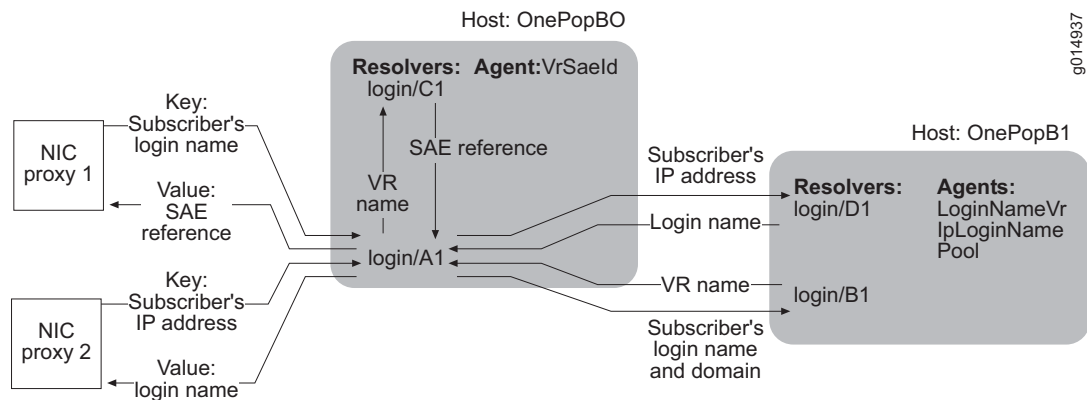
Figure 32: OnePopLogin Centralized Configuration



Distributed Configuration

In this configuration, the agents and resolvers are distributed among several hosts. When the NIC proxy sends a subscriber's IP address to the host OnePopBO, the resolvers execute the same actions as they do in the centralized configuration. Figure 33 illustrates the interactions of the NIC components for this realm.

Figure 33: OnePopLogin Distributed Configuration

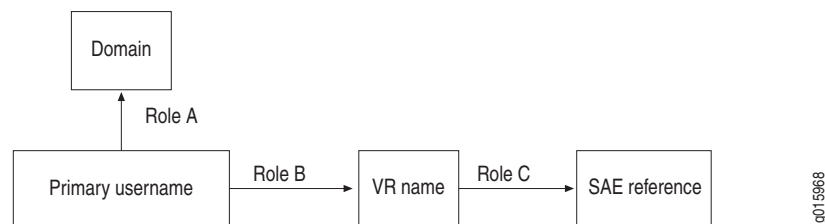


OnePopPrimaryUser

The OnePopPrimaryUser scenario is similar to one of the resolutions in the OnePopLogin scenario. In the OnePopPrimaryUser scenario, subscriber primary username, as identified by the PA_PRIMARY_USER_NAME attribute, is resolved to a reference for a managing SAE. The realm for this configuration accommodates a situation in which a NIC proxy provides a primary username.

Figure 34 show the resolution graph for this realm.

Figure 34: Resolution Processes for primary_user Realm



The following agents interact with resolvers in this realm:

- Directory agent VrSaeld collects and publishes information about virtual routers and the mappings between virtual routers and SAEs.
- SAE plug-in agent UserNameVr collects and publishes information about the mappings of subscriber primary usernames to VR names.

The OnePopPrimaryUser scenario provides two host configurations: a centralized configuration and a distributed configuration.

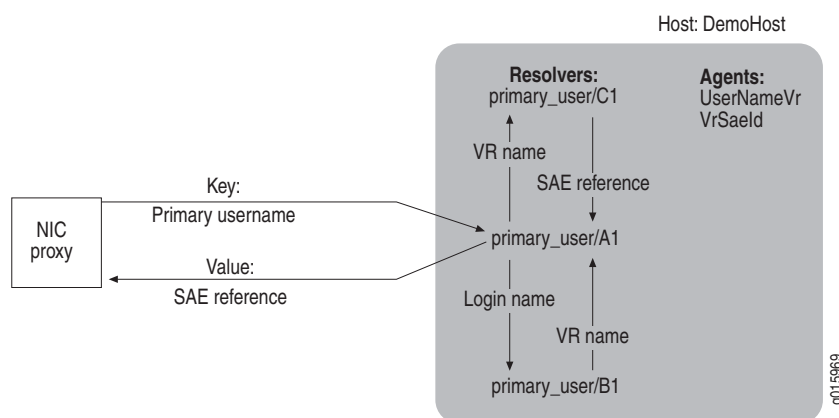
Centralized Configuration

In this configuration, a single host called DemoHost supports all agents and resolvers. When a NIC proxy send a subscriber's primary username to host Demo Host, the following sequence of events occurs:

1. The host passes the primary username to resolver A1.
2. (Optional) Resolver A1 resolves the primary username to its domain.
3. Resolver A1 forwards the primary username to resolver B1.
4. Resolver B1 obtains the name of the VR associated with the subscriber's primary username and returns the VR to resolver A1.
5. Resolver A1 forwards the VR to resolver C1.
6. Resolver C1 obtains the SAE reference for the SAE managing the VR and returns the SAE reference to resolver A1.
7. Resolver A1 returns the SAE reference to the host.
8. The host returns the SAE reference to the NIC proxy.

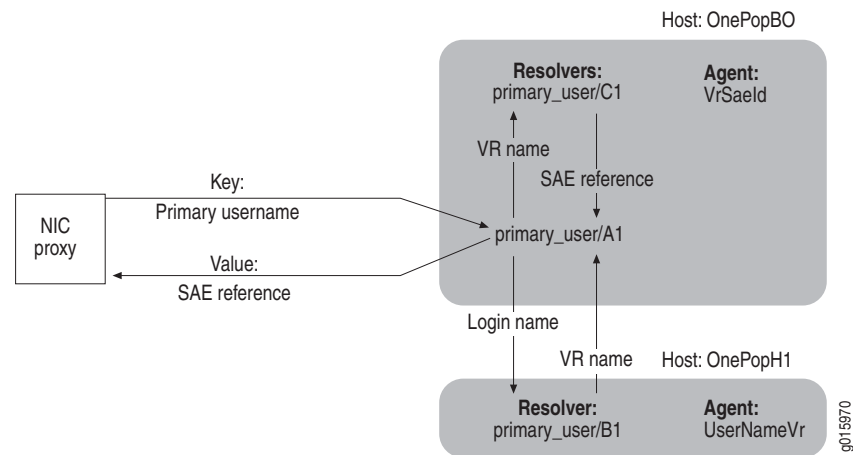
Figure 35 illustrates the interactions of the NIC components for this realm.

Figure 35: OnePopPrimaryUser Centralized Configuration



Distributed Configuration

In this configuration, the agents and resolvers are distributed among two hosts. When a NIC proxy sends a subscriber's primary username to the host OnePopBO, the resolvers execute the same actions as they do in the centralized configuration. Figure 36 illustrates the interactions of the NIC components for this realm.

Figure 36: OnePopPrimaryUser Distributed Configuration

OnePopDnSharedIp Scenario

The OnePopDnSharedIp scenario illustrates how to configure SAE plug-in agents that have state synchronization enabled to support an SAE plug-in that uses state synchronization. This scenario uses the same centralized and distributed configurations of hosts as the OnePop scenario.

Two realms are configured:

- **Shared IP**

The resolution process is identical to that for the OnePopShared scenario (see Figure 23 on page 336).

- **DN realm**

This realm uses essentially the same resolution process as the MultiPop DN realm (see Figure 44 on page 359). However, some of the constraints differ.

This realm also uses the same agents as the MultiPop DN realm. The names of agents and resolvers are essentially the same as those in the MultiPop configuration, although they do not include a POP identifier. Figure 37 on page 348 illustrates the centralized configuration, and Figure 38 on page 350 illustrates the distributed configuration for the DN realms.

The configuration for the two realms is similar to the configuration for the shared IP and DN realms in the OnePopAllRealms scenario. See *OnePopAllRealms Scenario* on page 351.

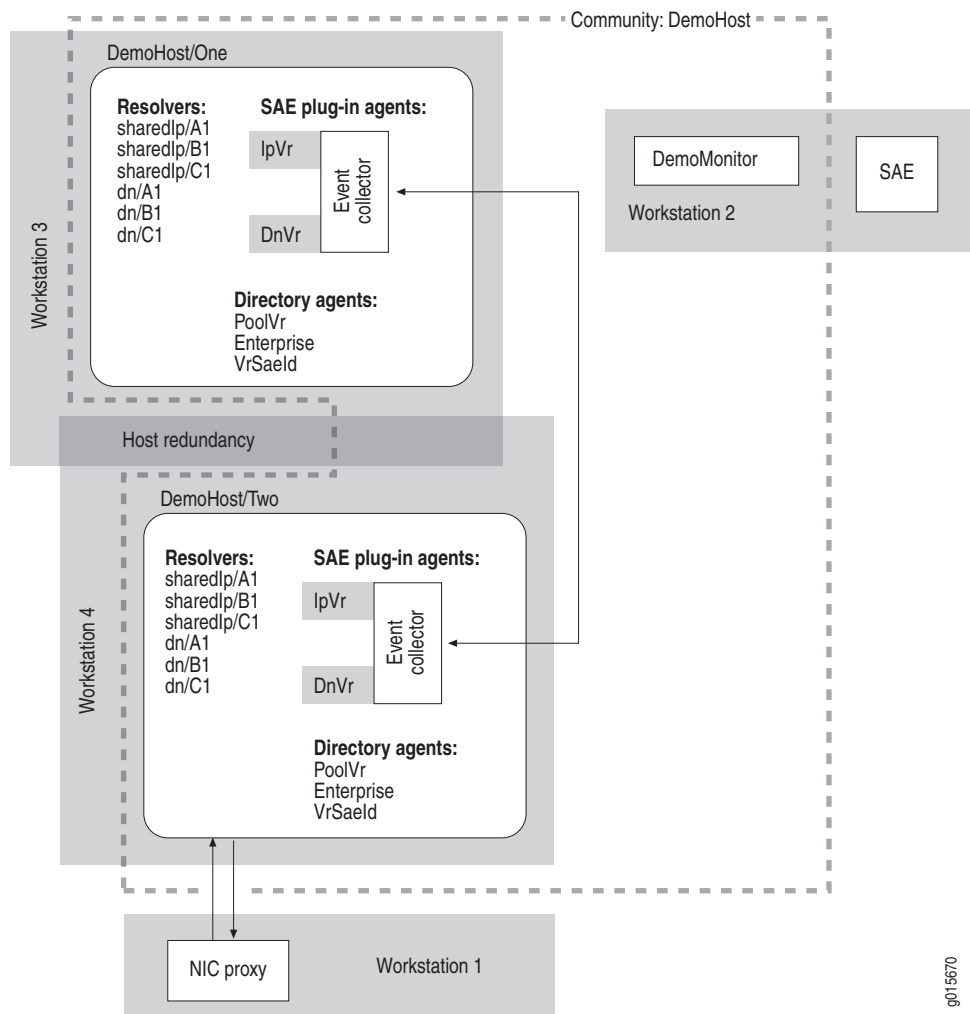
The OnePopAllRealms illustrates SAE plug-in agents configured to use SAE plug-in redundancy rather than SAE plug-in agents.

Centralized Configuration

Figure 37 on page 348 shows the centralized configuration for the scenario. Host DemoHost supports all resolvers and agents. The two SAE plug-in agents, IpVr and DnVr, share an event collector. Both plug-in agents have state synchronization enabled.

DemoHost is also configured for redundancy. Its redundant hosts (DemoHost/One and DemoHost/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host. The redundant hosts form a community called DemoHost with the monitor DemoMonitor, which tracks them.

Figure 37: OnePopDnSharedIp Realms Centralized Configuration



g015670

Distributed Configuration

Figure 38 on page 350 shows the distributed configuration from the scenario. Host OnePopBO supports two resolvers for each realm and a directory agent that is used by different realms. Host OnePopH1 supports one resolver for each realm and agents that are used by different realms.

Both hosts also have a redundant configuration. The redundant hosts for OnePopBO (OnePopBO/One and OnePopBO/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host.

The redundant hosts for OnePopH1 (OnePopH1/One and OnePopH1/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host.

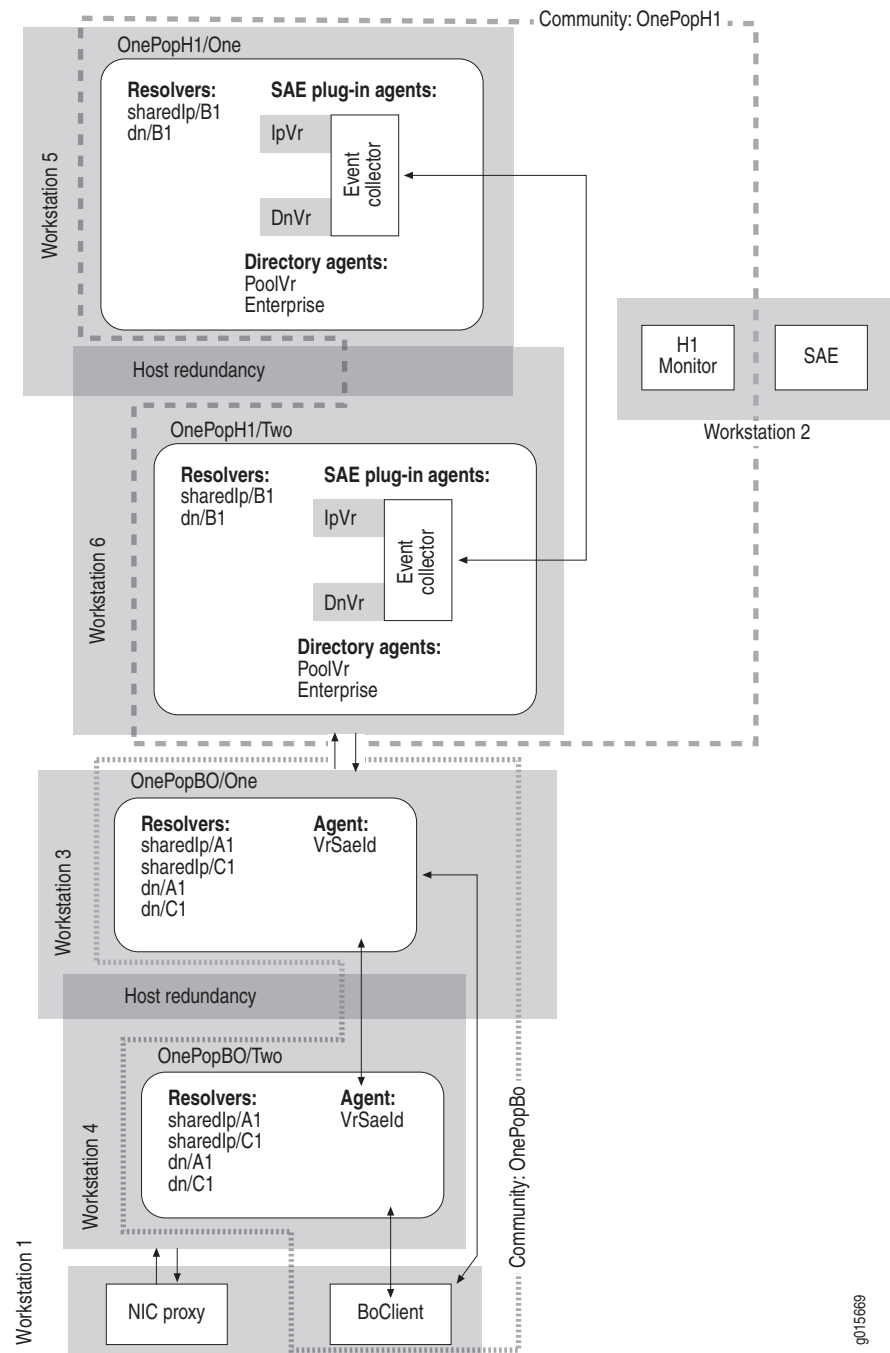
However, host OnePopH1 also supports two SAE plug-in agents, IpVr and DnVr, which share an event collector. These agents have state synchronization enabled.

The redundant hosts OnePopBO/One and OnePopBO/Two are members of a community called OnePopBO. This community supports the monitor, BoClient, which is installed on the machine that supports the NIC proxy. BoClient tracks the connections between the redundant hosts OnePopBO/One and OnePopBO/Two from the point of view of the NIC client (NIC proxy).

Similarly, the redundant hosts OnePopH1/One and OnePopH1/Two are members of a community called OnePopH1. This community has one monitor, H1Monitor, which is located on the same machine as the SAE and tracks the connections among the redundant hosts in the same community, their primary host, and the other hosts in the configuration.

H1Monitor comprises the monitor process OnePop, which is installed on the same machine as the SAE. BoClient comprises the monitor process OnePopClient, which is installed on the same machine as the NIC proxy.

Figure 38: OnePopDnSharedIp Realms Distributed Configuration



g015669

OnePopAllRealms Scenario

The main purpose of the OnePopAllRealms scenario is to illustrate how to configure redundancy. This scenario uses the same centralized and distributed configurations of hosts as the OnePop scenario.

Three realms are configured:

- IP realm

This realm uses essentially the same resolution process as the IP realm for the OnePop scenario (see Figure 13 on page 328). However, some of the constraints differ.

- Shared IP

The resolution process is identical to that for the OnePopShared scenario (see Figure 23 on page 336).

- DN realm

This realm uses essentially the same resolution process as the MultiPop DN realm (see Figure 44 on page 359). However, some of the constraints differ.

This realm also uses the same agents as the MultiPop DN realm. The names of agents and resolvers are essentially the same as those in the MultiPop configuration, although they do not include a POP identifier. By reviewing the scenario, Figure 39 and Figure 40, you can determine exact pictures of the DN realms for the centralized and distributed configurations.

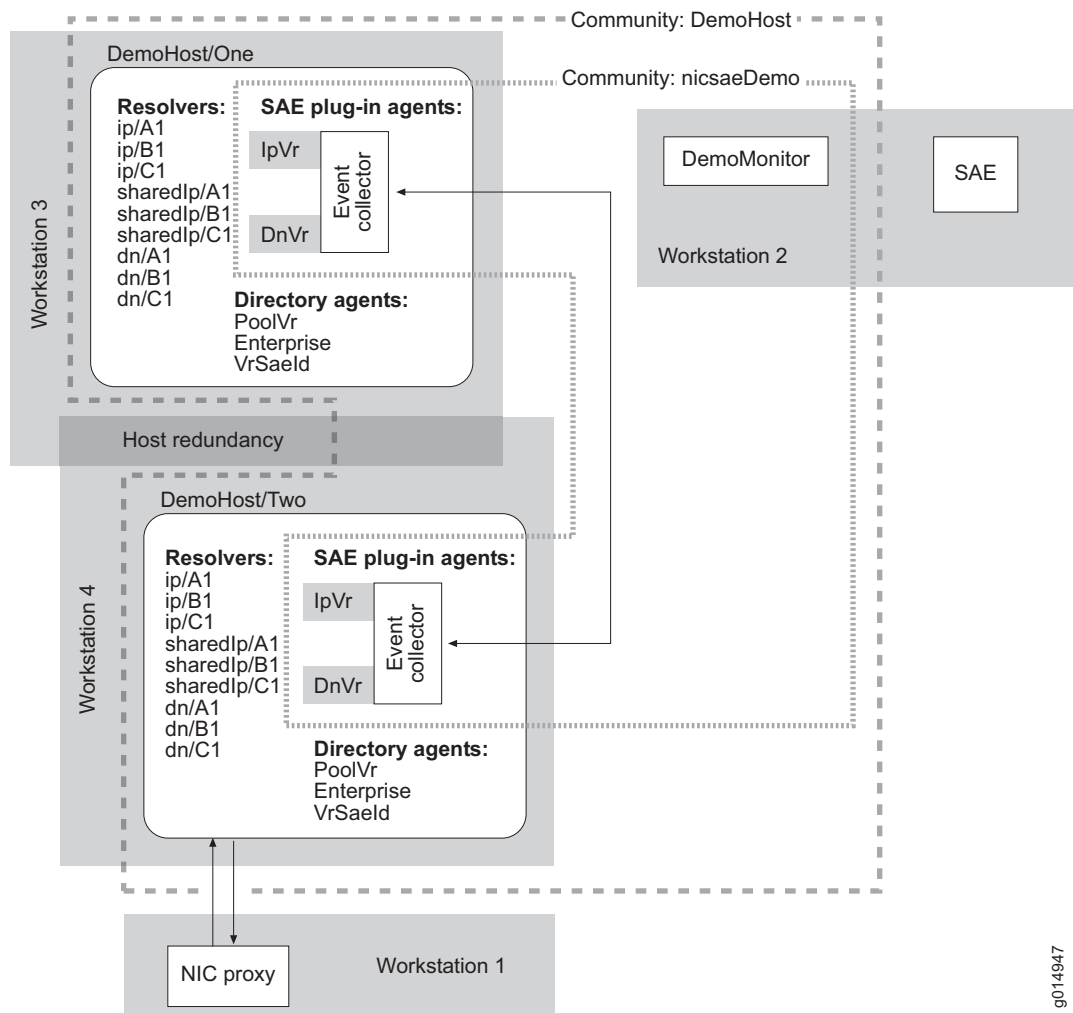
Centralized Configuration

Figure 39 on page 352 shows the centralized configuration for the scenario. Host DemoHost supports all resolvers and agents. However, because host DemoHost is configured for redundancy, its redundant hosts (DemoHost/One and DemoHost/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host.

The parent host DemoHost also supports two SAE plug-in agents, IpVr and DnVr, which share an event collector. Each SAE plug-in agent has a redundant agent called Demo; these redundant agents also share an event collector. The redundant agents and their shared event collector are assigned to both redundant hosts DemoHost/One and DemoHost/Two.

The redundant agents form a community called nicsaeDemo with the monitor DemoMonitor, which tracks them. The redundant agents are identified in the community by the names DemoHost/One and DemoHost/Two; these names specify their hosts and provide unique identifiers for the redundant agents.

The redundant hosts form a community called DemoHost with the monitor DemoMonitor, which tracks them.

Figure 39: OnePopAllRealms Centralized Configuration

Distributed Configuration

Figure 40 on page 354 shows the distributed configuration for the scenario. Host OnePopBO supports two resolvers for each realm and a directory agent that is used by different realms. However, because host OnePopBO is configured for redundancy, its redundant hosts (OnePopBO/One and OnePopBO/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host.

Host OnePopH1 supports one resolver for each realm and agents that are used by different realms. Host OnePopH1 is also configured for redundancy, and its redundant hosts (OnePopH1/One and OnePopH1/Two) perform the host function. The redundant hosts are on different machines, and both hosts support the resolvers and agents assigned to the parent host.

However, host OnePopH1 also supports two SAE plug-in agents, IpVr and DnVr, which share an event collector. Each SAE plug-in agent has a redundant agent called onePop; these redundant agents also share an event collector. The redundant agents and their shared event collector are assigned to redundant hosts OnePopH1/One and OnePopH1/Two.

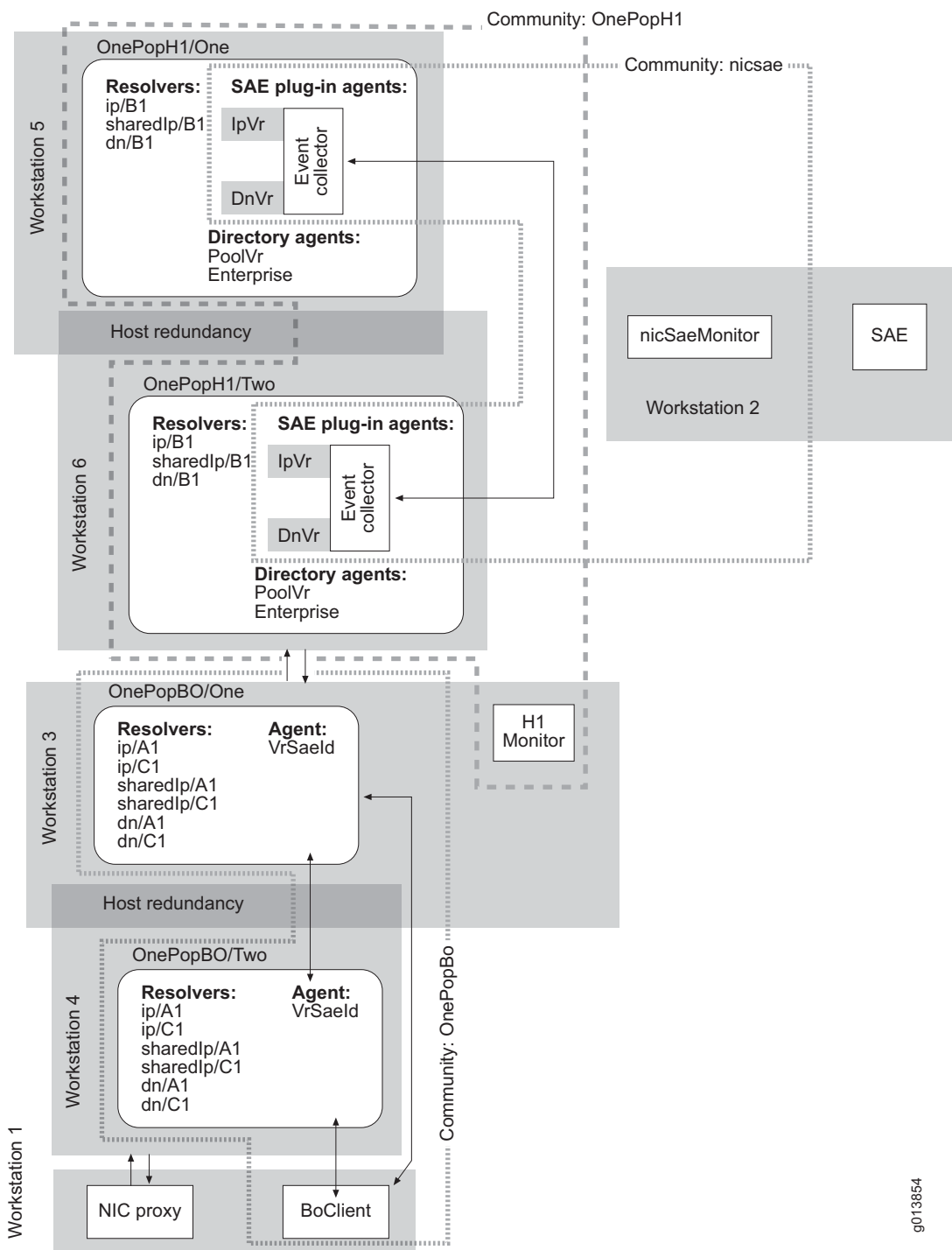
The redundant agents form a community called nicsae with monitor nicSaeMonitor, which tracks them. The redundant agents are identified in the community by the names OnePopH1/One and OnePopH1/Two; these names specify their hosts and provide unique identifiers for the redundant agents.

The redundant hosts OnePopBO/One and OnePopBO/Two are members of a community called OnePopBO. This community supports the monitor, BoClient, which is installed on the machine that supports the NIC proxy. BoClient tracks the connections between the redundant hosts OnePopBO/One and OnePopBO/Two from the point of view of the NIC client (NIC proxy).

Similarly, the redundant hosts OnePopH1/One and OnePopH1/Two are members of a community called OnePopH1. This community has one monitor, H1Monitor, which is located on the same machine as the SAE and tracks the connections among the redundant hosts in the same community, their primary host, and the other hosts in the configuration.

H1Monitor and nicSaeMonitor are part of the monitor process OnePop, which is also installed on the same machine as the SAE. BoClient is part of the monitor process OnePopClient, which is installed on the same machine as the NIC proxy.

Figure 40: OnePopAllRealms Distributed Configuration



g013854

MultiPop Scenario

The MultiPop scenario illustrates a configuration that involves two POPs: Montreal and Ottawa. This configuration does not provide redundancy. The NIC proxy communicates with the back office host (BackOffice), which in turn communicates with the POP hosts (MontrealHost and OttawaHost). Hosts MontrealHost and OttawaHost support equivalent hosts and agents and manage resolutions in the same way.

When host BackOffice receives a data key from the NIC proxy, the following sequence of events occurs:

1. Host BackOffice forwards requests as follows:
 - If the request is for the Montreal POP, host BackOffice forwards the request to POP host MontrealHost.
 - If the request is for the Ottawa POP, host BackOffice forwards the request to POP host OttawaHost.
2. Delegating tasks to other resolvers as necessary, the resolvers in the POP obtain data values that correspond to the data key request, and return them.
3. The POP host returns the data values to host BackOffice, which returns the value to the NIC proxy.

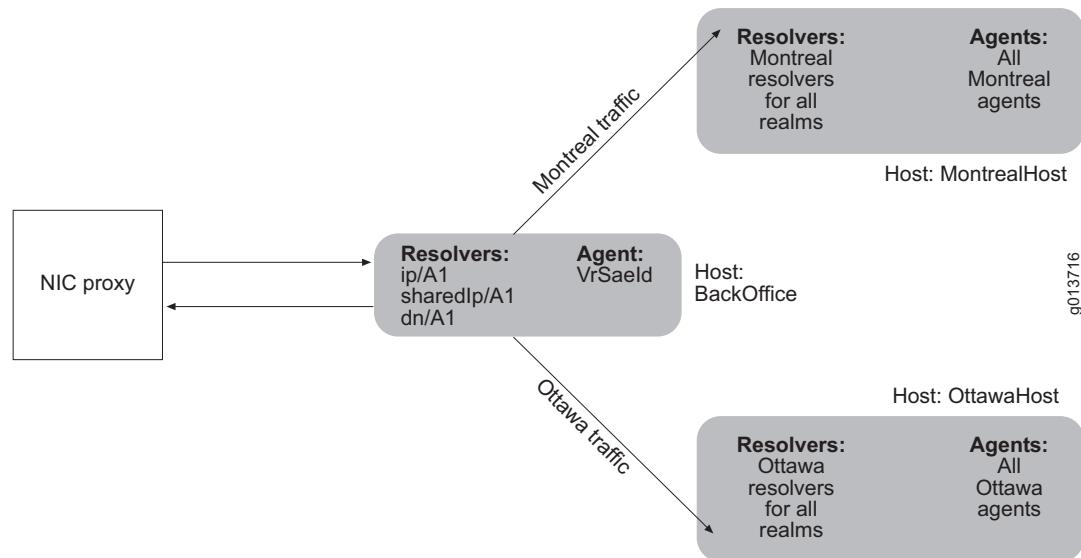
The scenario shows three realms for this configuration:

- IP
- Shared IP
- DN

Each realm provides a different type of resolution. The following sections provide information about these realms.

Figure 41 illustrates this configuration.

Figure 41: MultiPop Configuration



IP Realm

This realm accommodates the situation in which IP address pools are configured locally on each VR. The resolution process takes a subscriber's IP address as the key and returns a reference to the SAE managing this subscriber as the value. This realm uses essentially the same resolution process as the ip realm for the OnePop scenario (see Figure 13 on page 328). However, some of the constraints differ.

The following agents interact with the resolvers in this realm:

- Directory agents **montrealPoolVr** and **ottawaPoolVr** collect and publish information that maps IP address pools to VRs. Each agent publishes only the information that is relevant to its POP. You achieve selective publishing by relating an Ottawa scope to the VRs in the Ottawa POP and a Montreal scope to the VRs in the Montreal POP and defining a search filter for the agents to load only the VRs in its POP.
- Directory agent **VrSaeld** in the back office collects and publishes information that maps VRs to SAEs for both POPs.

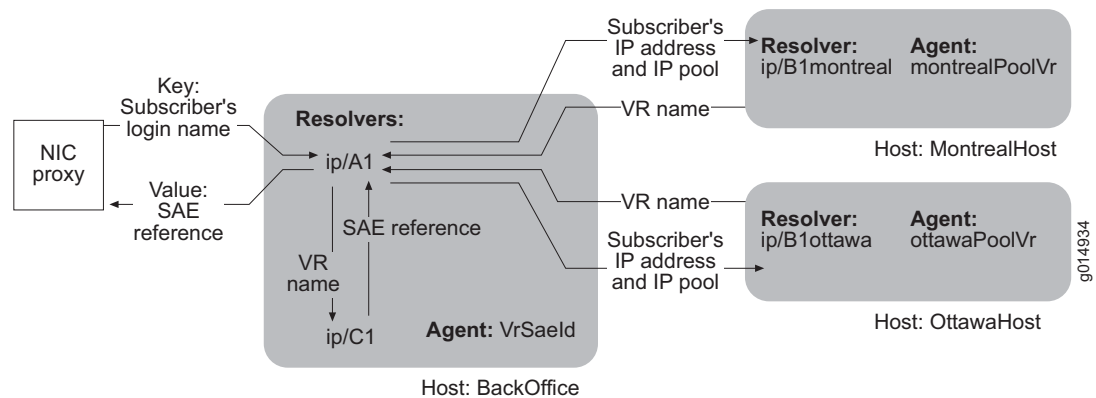
When the NIC proxy sends a subscriber's IP address to host BackOffice, the following sequence of events occurs:

1. Host BackOffice passes the IP address to resolver ip/A1.
2. Resolver ip/A1 obtains an IP pool for the IP address.
3. Resolver ip/A1, based on the value of the IpPool, forwards the request to ip/B1montreal or ip/B1ottawa.

4. Resolver ip/B1montreal or resolver ip/B1ottawa obtains a VR name for this IP pool and returns the VR name to resolver ip/A1.
5. Resolver ip/A1 forwards the VR name to resolver ip/C1.
6. Resolver ip/C1 obtains the SAE identity for this VR and returns the value to resolver ip/A1.
7. Resolver ip/A1 returns the SAE reference to its host.
8. Host BackOffice returns the SAE reference to the NIC proxy.

Figure 42 illustrates the interactions of the NIC components for this realm.

Figure 42: iP Realm for MultiPop Configuration



Shared IP Realm

This realm accommodates the situation in which IP address pools are shared by VRs in the same POP. The realm takes a subscriber's IP address as the key and returns the corresponding SAE as the value. Figure 14 on page 329 shows the resolution graph for this realm.

The following agents interact with resolvers in this realm:

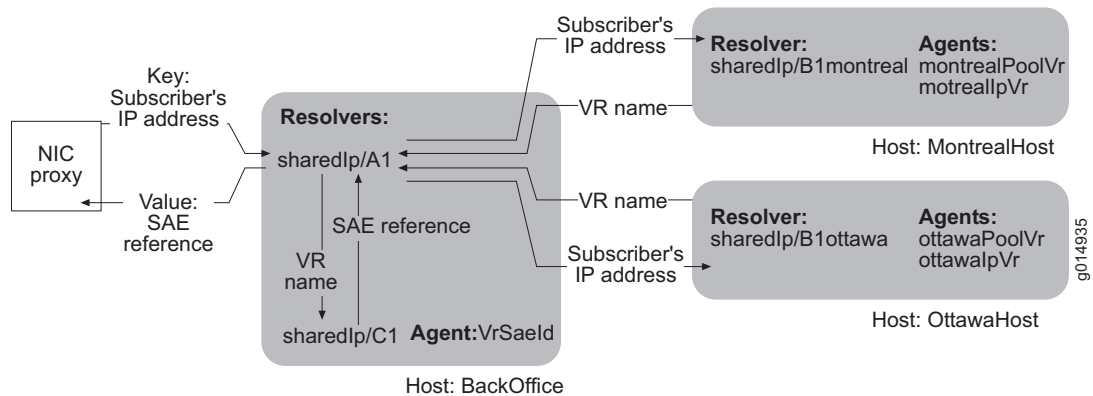
- Directory agents montrealPoolVr and ottawaPoolVr collect and publish information about the mappings of IP address pools to VRs. Each agent publishes only the information that is relevant to its POP.
- SAE plug-in agents montrealIpVr and ottawaIpVr collect and publish information about the mappings of subscriber IP addresses to VRs. Each agent publishes only the information that is relevant to its POP.
- Directory agent VrSaeld in the back office collects and publishes information about the mappings of VRs to SAEs for both POPs.

When the NIC proxy sends a subscriber's IP address to host BackOffice, the following sequence of events occurs:

1. Host BackOffice passes the IP address to resolver sharedIp/A1.
2. Resolver sharedIp/A1 obtains an IP pool for the IP address.
3. Resolver sharedIp/A1, based on the value of the IP pool, forwards the request to sharedIp/B1montreal or sharedIp/B1ottawa.
4. Resolver sharedIp/B1montreal or resolver sharedIp/B1ottawa obtains a VR name for this IP address and returns the VR name to resolver sharedIp/A1.
5. Resolver sharedIp/A1 forwards the VR name to resolver sharedIp/C1.
6. Resolver sharedIp/C1 obtains the SAE identity for this VR and returns the value to resolver sharedIp/A1.
7. Resolver sharedIp/A1 passes the SAE reference to its host.
8. Host BackOffice returns the SAE reference to the NIC proxy.

Figure 43 illustrates the interactions of the NIC components for this realm.

Figure 43: sharedIP Realm for MultiPop Configuration

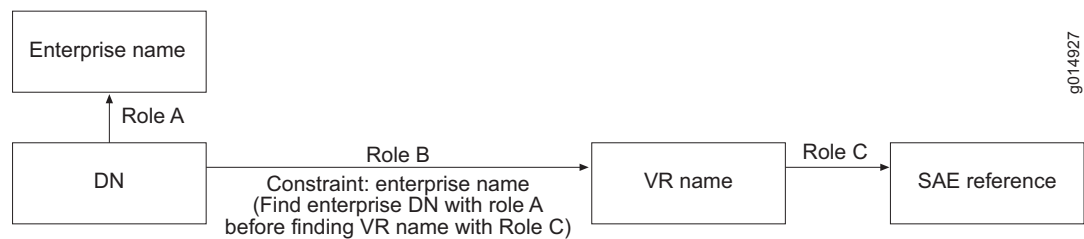


DN Realm

The DN realm takes the DN of an access subscriber (an access DN) as the key and returns the corresponding SAE as the value. Figure 44 shows the resolution process for this realm.

Figure 44 shows the resolution graph for this realm.

Figure 44: Resolution Graph for MultiPOP dn Realm



The following agents interact with resolvers in this realm:

- Directory agents ottawaEnterprise and montrealEnterprise collect and publish information about the DNs of enterprise subscribers (enterprise DNs). Each agent publishes only the information that is relevant to its POP. You achieve selective publishing by relating an Ottawa service scope to the enterprises in the Ottawa POP and a Montreal service scope to the enterprises in the Montreal POP and defining a search filter for the agents to load only the enterprises in its POP.
- SAE plug-in agents montrealDnVr and ottawaDnVr collect and publish information about the mappings of access DNs to VRs. Each agent publishes only the information that is relevant to its POP.
- Directory agent VrSaeld collects and publishes information about the mappings of VRs to SAEs for both POPs.

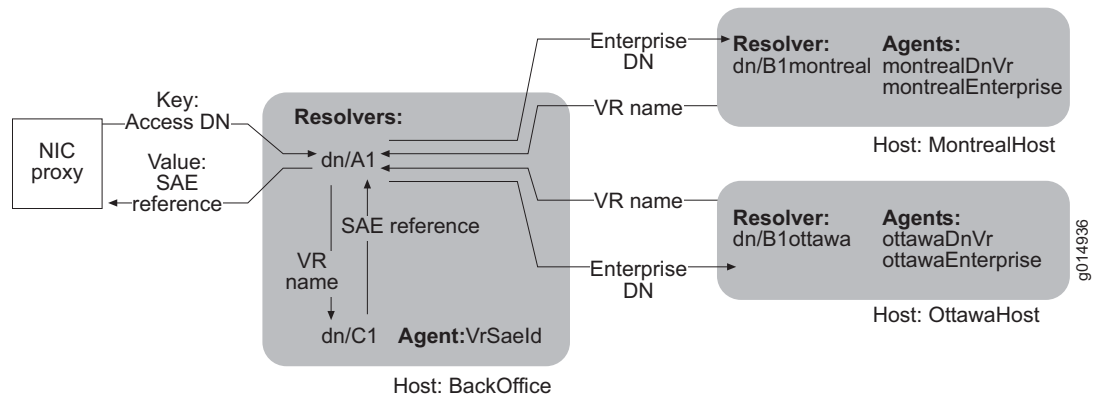
When the NIC proxy sends an access DN to host BackOffice, the following sequence of events occurs:

1. Host BackOffice passes the access DN to resolver dn/A1.
2. Resolver dn/A1 obtains an enterprise DN for the access DN.
3. Resolver dn/A1, based on the value of the enterprise DN, forwards the request to dn/B1montreal or dn/B1ottawa.
4. Resolver dn/B1montreal or resolver dn/B1ottawa obtains a VR name for this enterprise DN and returns the VR name to resolver dn/A1.
5. Resolver dn/A1 forwards the VR name to resolver dn/C1.
6. Resolver dn/C1 obtains the SAE reference for this VR and returns the value to resolver dn/A1.

7. Resolver dn/A1 passes the SAE reference to its host.
8. Host BackOffice returns the SAE reference to the NIC proxy.

Figure 45 illustrates the interactions of the NIC components for this realm.

Figure 45: dn Realm for MultiPop Configuration



Index

A

access DNs 283
 accounting
 SAE, description 10
 address pools. *See* IP address pools 62, 86
 agents
 See also NIC
 APIs (application programming interfaces)
 CORBA remote API 9
 NIC 263
 provided with SAE 8
 SAE core API 8
 application programming interfaces. *See* APIs
 applications
 SRC on CD xix
 audience for documentation xvii

B

BEEP, JUNOS routing platforms 4
 configuring port
 SDX Configuration Editor 145
 SRC CLI 114
 connection 110, 136
 Blocks Extensible Exchange Protocol. *See* BEEP

C

certificate authority (CA) 115, 148
 component interactions
 JUNOS routing platforms and SAE 4
 configuration group, JUNOS
 routing platforms 110, 127, 136, 163
 configuration manager, instantiating for NIC 264
 conventions defined
 icons xviii
 text xviii
 COPS (Common Open Policy Service)
 connection with JUNOSe routers 60, 82
 configuring SAE, SDX Configuration Editor 90
 configuring SAE, SRC CLI 64
 disabling on router 73, 106
 enabling on router 72, 105
 COPS-PR versus COPS XDR 4
 JUNOSe router connection 4

CORBA (Common Object Request Broker Architecture)

 IOR location 242
 naming server for NIC 185
 reference for SAE
 JUNOS 142
 JUNOSe 88
 remote API 9

CORBA-based plug-in SPI. *See* plug-ins, external

customer support xxii
 customized interface modules 9

D

device drivers

 JUNOS

 configuring, SDX Configuration Editor 144
 configuring, SRC CLI 113
 viewing state, C-Web interface 132
 viewing state, SRC CLI 130
 viewing statistics, C-Web interface 133, 134
 viewing statistics, SRC CLI 130, 131

 JUNOSe

 configuring, SDX Configuration Editor 90
 configuring, SRC CLI 64
 viewing state, C-Web interface 77
 viewing state, SRC CLI 75
 viewing statistics, C-Web interface 78, 79
 viewing statistics, SRC CLI 76

directory

 multimaster replication 272
 directory blacklist, deleting 53

distinguished name. *See* DN

DN (distinguished name)

 NIC resolution 283

documentation set, SRC. *See* SRC documentation set

domain maps

 reloading on SAE 53

E

equipment registration

 deleting 54

external applications, interaction with NIC 263

external plug-ins. *See* plug-ins

F

failover parameters, SAE 55

G

groups, NIC hosts 179, 222

Hhosted plug-ins. *See* plug-ins

hosts

See also NIC hosts**I**

icons defined, notice xviii

interface classification scripts

reloading on SAE 53

interface modules, SAE 9

internal plug-ins. *See* plug-ins

IOR

managing SAE

JUNOS, SDX Admin 142

JUNOSe, SDX Admin 88

router initialization scripts 68, 95, 123, 157

IP address pools

local address pools, configuring

SDX Admin 86

SRC CLI 62

static pools, configuring

SDX Admin 87

SRC CLI 63

updating for JUNOSe VRs 99–102

poolRepublish 100

SDX Admin 100

J

JacORB, for NIC

configuring

default ORB for JRE 251

properties on redundant NIC hosts 278

Web application 252

Web application server 253

JRE package 251

Java garbage collection, running 53

Java Management Extension 258

Java Runtime Environment, NIC proxy 251

Java Virtual Machine, for NIC proxy 251

JMX (Java Management Extension) software 258

JUNOS routing platforms

accessing router CLI 126, 160

BEEP connection 4

configuring port, SDX Configuration Editor 145

configuring port, SRC CLI 114

configuration groups 110, 127, 136, 163

configuring to interact with SAE 126, 162

default virtual router 110, 136

disabling interactions with SAE 128, 163

enabling interactions with SAE 128, 163

monitoring interactions with SAE 128, 164

router objects, adding

SDX Admin 137–139

SRC CLI 111

SAE interactions 4

SRC software process 110, 136

statements for integration

port-number 127, 162

server-address 127, 162

source-address 127, 163

troubleshooting 129, 164

VR objects, adding individually

SDX Admin 139–144

SRC CLI 112

JUNOSe routers

accessing router CLI 72, 103

COPS connection 4

configuring, SDX Configuration Editor 90

configuring, SRC CLI 64

integration overview 60, 82

monitoring interactions with SAE 73, 106

router objects, adding

SDX Admin 82

SRC CLI 60

SNMP communities, configuring

SDX Configuration Editor 94

SRC CLI 67

SNMP server

configuring on router 66, 93

SRC client 60, 82

starting 72, 105

stopping 73, 106

troubleshooting 74, 107

VR objects

adding individually, SDX Admin 85–90

adding individually, SRC CLI 62

discovering, SDX Admin 82

discovering, SRC CLI 60

JVM (Java virtual machine), for NIC proxy 251

LLDAP access. *See* SAE (service activation engine), configuring

login names 283

login registration

deleting 54

M

manuals, SRC

comments xxii

monitors

configuring NIC 277

multimaster directory replication 272

Nnetwork information collector. *See* NICnetwork publisher. *See* NIC 224

NIC (network information collector)

API 263

communities 271, 274

configuration prerequisites

C-series platforms 190

Solaris platforms 208

configuration statements 188

configuration, changing 204

configuration, verifying 203

- CORBA naming server..... 185
- data mapping 177, 324
- default operating properties, viewing 192
- factory interface 263, 264
- log files 282
- logging
 - changing configuration 196
 - default 196
- monitors 272
 - configuring 277
 - example 353
 - starting 279
 - stopping 280
- network publisher
 - configuration file 224
 - input directory 230
 - output file 230
 - overview 224
 - prerequisites 224
 - running 229
 - troubleshooting 229
- operating properties, changing 192
- overview 175
- planning implementation 181
- realms
 - configuring 298
 - overview 284
- replication
 - configuring 222
 - groups 179, 212, 222
 - overview 179
 - SAE plug-in agents 201, 219
- resolution performance 247
- resolution processes 283, 284
- resolvers
 - constraints 287
 - overview 178
 - viewing data mappings 324
- restarting 204
- roles 284
- starting 191, 279
- stopping 203
- testing 203, 253
 - any key 242
 - test data 241
- troubleshooting resolution processes 324
- viewing
 - configuration 328
 - host log 282
 - host status 222
 - monitor log 282
 - monitor status 280
- See also other NIC entries*
- NIC agents
 - configuration overview 184
 - consolidator 300, 301
 - consolidator agent 301
 - directory, changes, managing 295
 - directory, configuring
 - SRC CLI 197
 - directory, reviewing configuration 303
 - overview 178
 - properties 313, 314
 - restarting 204
 - SAE plug-in
 - configuration, reviewing 307, 309
 - sae plug-in agents, configuring 199
 - XML 316, 317, 323
- NIC configuration scenarios
 - changing 205, 222
 - SRC CLI 195
 - Multipop 355
 - OnePop 328
 - OnePopAcctId 340
 - OnePopAllRealms 351
 - OnePopDnSharedIp 347
 - OnePopDynamicIp 334
 - OnePopLogin 342
 - OnePopPcmm 332
 - OnePopPrimaryUser 345
 - OnePopSharedIp 336
 - OnePopStaticRoutelp 223, 338
 - OnePopStatisRoutelp 202
 - overview 182, 328
 - static-dn 205
- NIC hosts
 - configuration prerequisites 190, 208
 - configuring 208–222, 321, 322
 - groups 179, 212, 222
 - logging 322
 - overview 178
 - redundancy
 - configuring 271–279
 - configuring JacORB properties 278
 - example 351–354
 - overview 271
 - starting 191, 221
 - stopping 203, 204, 221
- NIC locators
 - configuration 323
 - external applications 261, 262
 - overview 176, 177
- NIC proxies
 - cache, configuring
 - SDX Configuration Editor 247
 - SRC CLI 238
 - configuration overview 233
 - configuration prerequisites 236
 - instantiating 266
 - JacORB 251
 - logging 265
 - monitoring 258
- NIC replication, configuring
 - SDX Configuration Editor 249
 - SRC CLI 239
- optimizing performance 280
- ORB version 251
- overview 177

- prerequisites 234
 - proxy stub 255
 - removing instances 268
 - replication 244, 249, 255
 - requirements 263
 - resetting MBeans 259
 - resolution information, configuring
 - SDX Configuration Editor 245
 - SRC CLI 237
 - resolution requests 266
 - NIC resolvers
 - restarting 204
 - nicDump program 324
 - notice icons defined xviii
- O**
- objectives of guide xvii
 - ORB (object request broker), NIC proxy 280
- P**
- PacketCable Multimedia. *See* PCMM
 - PCMM (PacketCable Multimedia)
 - SAE connection 5
 - performance
 - NIC proxy 280
 - plug-ins
 - architecture 6
 - external 7
 - hosted 7
 - internal 6
 - tracking
 - virtual routers, SDX Admin 88, 143
 - virtual routers, SRC CLI 63, 113
 - types 6
 - Policy Editor
 - accessing routers 103, 160
 - poolRepublish command
 - executing 100
 - prerequisites 99
 - troubleshooting 102
 - variables 101
 - priorityList 240
- R**
- randomPick 240
 - realm
 - See* NIC realms
 - release notes xxi
 - resolution processes
 - DN to SAE reference 347, 351, 359
 - IP address to login name 342
 - IP address to SAE reference 328, 336, 347, 351, 356
 - login name to SAE reference 342
 - roles, NIC 284
 - roundRobin 240
 - router initialization scripts
 - iorPublisher 68, 95
 - JUNOS 123, 157
 - configuring location, SDX Configuration Editor 159
 - configuring location, SRC CLI 125
 - example 125, 158
 - JUNOSe 68, 95
 - configuring location, SDX Configuration Editor 98
 - configuring location, SRC CLI 70
 - example 70, 97
 - poolPublisher 68, 95
 - specifying for NIC 186, 208
 - routers
 - accessing router CLI 72, 103
 - adding JUNOS routing platforms
 - SDX Admin 136–144
 - SRC CLI 110
 - adding JUNOSe
 - SDX Admin 82–90
 - SRC CLI 60
 - integrating JUNOS routing platform 110, 136
 - integrating JUNOSe 60, 82
- S**
- SAE (service activation engine)
 - accounting 10
 - APIs. *See* APIs
 - BEEP connection, JUNOS routing platforms 4
 - COPS
 - JUNOSe router connection 4
 - deleting directory blacklist 53
 - disabling interactions with JUNOS routing platform 128, 163
 - enabling interactions with JUNOS routing platform 128, 163
 - failover parameters 55
 - JUNOS routing platform client 126, 162
 - monitoring interactions
 - JUNOS routing platform 128, 164
 - JUNOSe routers 73, 106
 - NIC replication, configuring
 - SDX Configuration Editor 219
 - SRC CLI 201
 - overview 3–12
 - PCMM environment 5
 - plug-ins *See* plug-ins
 - reloading configuration 52
 - role 3
 - router initialization scripts. *See* router initialization scripts
 - session store
 - C-series platforms 22
 - Solaris platforms 41
 - starting
 - SRC client on JUNOSe router 72, 105
 - stopping
 - SRC client on JUNOSe router 73, 106

- SAE (service activation engine), configuring
 - BEEP connection
 - SDX Configuration Editor 145
 - SRC CLI 114
 - COPS connection
 - SDX Configuration Editor 90
 - SRC CLI 64
 - directory eventing, SAE configuration data
 - SDX Configuration Editor 40
 - SRC CLI 20
 - LDAP access, SDX Configuration Editor
 - device data 40
 - directory data 29
 - persistent login cache data 37
 - persistent session cache repository 40
 - policy data 35
 - service data 33
 - subscriber data 30
 - LDAP access, SRC CLI
 - device data 20
 - directory data 13
 - persistent login cache data 18
 - policy data 17
 - service data 16
 - subscriber data 14
 - router initialization script location
 - SDX Configuration Editor 98, 159
 - SRC CLI 70, 125
 - serialized data compression
 - SDX Configuration Editor 48
 - SRC CLI 27
 - session job manager
 - SDX Configuration Editor 50
 - SRC CLI 28
 - session store
 - SDX Configuration Editor 43
 - SRC CLI 23
 - SNMP communities
 - C-series platforms 66
 - Solaris Platforms 93
 - IOR, JUNOS 142
 - IOR, JUNOSe 88
 - See also* SAE (service activation engine)
 - SAE remote interface
 - customized interface modules 9
 - serialized data compression, configuring
 - SDX Configuration Editor 48
 - SRC CLI 27
 - service activation engine. *See* SAE
 - services
 - reloading on SAE 52
 - session job manager, configuring
 - SDX Configuration Editor 50
 - SRC CLI 28
 - session store
 - configuring, SDX Configuration Editor
 - compressing session objects 48
 - device driver 43
 - global parameters 47
 - configuring, SRC CLI
 - compressing session objects 27
 - device driver 23
 - global parameters 26
 - C-series platforms 22
 - Solaris platforms 41
 - SNMP
 - configuring
 - server on JUNOSe router 93
 - SNMP communities
 - configuring on SAE
 - SDX Configuration Editor 93
 - SRC CLI 66
 - SNMP server
 - configuring on JUNOSe router 66
 - SRC client, JUNOSe routers
 - configuring 60, 82
 - starting 72, 105
 - stopping 73, 106
 - SRC documentation set
 - comments xxii
 - obtaining xxi
 - SRC documentation CD xix
 - SRC software distribution xxi
 - SRC software process, JUNOS routing platforms 110, 136
 - disabling 128, 163
 - reenabling 128, 163
 - subscribers
 - IP addresses 283
 - login names 283
 - subscriptions
 - reloading on SAE 53
 - support, requesting xxii
- T**
- technical support, requesting xxii
 - testing
 - components that use NIC 253
 - text conventions defined xviii
 - threads
 - configuring for COPS 93
 - configuring for sessions
 - SDX Configuration Editor 50
 - SRC CLI 28
 - tracking plug-ins
 - virtual routers
 - SDX Admin 88
 - SRC CLI 63, 113
 - virtual routers, SDX Admin 143
 - troubleshooting
 - JUNOS routing platforms 129, 164
 - JUNOSe routers 74, 107
 - poolRepublish command 102

V

virtual routers

adding individually for JUNOS routing platforms	112
SDX Admin	139–144
adding individually for JUNOSe routers	
SDX Admin	85
SRC CLI	62
adding operative VRs	111
SDX Admin	82
SRC CLI	60
updating IP address pools. <i>See</i> IP address pools	99